Proyecto Fin de Grado
Grado en Ingeniería en Tecnologías Industriales

Simulación Ray-Tracing de flujo óptico para paneles solares e integración en MATLAB para control

Autor: Andrés Porras Salguero
Tutor: Carlos Vivas Venegas

Dep. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2018

Proyecto Fin de Grado
Ingeniería en Tecnologías Indistriales

Simulación Ray-Tracing de flujo óptico para paneles solares e integración en MATLAB para control

Autor:
Andrés Porras Salguero

Tutor:
Carlos Vivas Venegas
Profesor titular

Dep. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2018

Proyecto Fin de Grado: Simulación Ray-Tracing de flujo óptico para paneles solares e integración en MATLAB para control

Autor: Andrés Porras Salguero
Tutor: Carlos Vivas Venegas

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2018

El Secretario del Tribunal

Agradecimientos

A mis padres
A mi familia
A mis amigos

Resumen

El objetivo de este proyecto es realizar un análisis de la eficiencia de una célula fotovoltaica empleando técnicas de trazado de rayos, o Ray Tracing, como se denomina comúnmente en inglés. En este tipo de técnicas se simula la trayectoria seguida por un gran número de rayos de luz desde que se generan en la fuente hasta que inciden en la o las superficies de interés. En el caso que nos ocupa, se simula el camino seguido por los rayos solares en el interior de celda fotovoltaica considerando la absorción y reflejos de las diferentes superficies que la componen, hasta que inciden en el sustrato fotovoltaico.

Este tipo de técnicas permiten calcular, conocida la geometría de una celda, así como la reflectancia, absorbancia y transmitancia ópticas de las superficies que la componen, las propiedades de interés, concretamente la densidad de corriente fotogenerada y la eficiencia energética de la misma.

El código de simulación se ha desarrollado en MATLAB tomando como referencia para verificación y contraste de resultados el software desarrollado para propósito similar por la empresa PV Lighthouse Pty. Ltd.

El objetivo último del proyecto es desarrollar una herramienta software flexible y adaptable que pueda ser integrada con los paquetes de optimización numérica de MATLAB. Se espera que esta herramienta permita en futuras ampliaciones de este trabajo, acometer tareas de diseño optimizado de celdas fotovoltaicas de alta concentración (HCPV) con óptica móvil.

Abstract

The objective of this project is to perform an analysis of the efficiency of a photovoltaic cell using ray tracing techniques. In this type of techniques, the trajectory followed by a large number of light rays is simulated, following their paths from the source until they impinge the target surface. In the case at hand, the paths followed by the solar rays inside the photovoltaic cell are simulated, considering the absorption and reflections of the different surfaces, until they hit the photovoltaic substrate.

This type of technique allows to calculate, known the geometry of a cell, as well as the optical reflectance, absorbance and transmittance of the surfaces that compose it, the properties of interest, specifically the photogenerated current density and the energy efficiency.

The simulation code has been developed in MATLAB taking as a reference for verification and assessment the software developed for similar purpose by the company PV Lighthouse Pty. Ltd.

The ultimate goal of the project is to develop a flexible and adaptable software tool that can be integrated with numerical optimization packages of MATLAB. It is expected that future extensions of this work this tool will allow to undertake tasks of optimized design of high concentration photovoltaic cells (HCPV) with mobile optics.

Índice

1 INTRODUCCIÓN

1.1 Objetivos del Proyecto

El objetivo de este proyecto es estudiar el comportamiento de una célula fotovoltaica mediante la técnica de trazado de rayos o "Ray Tracing". Este procedimiento se basa en estudiar la trayectoria de un gran número de rayos de luz desde la fuente que los origina, en su evolución por el interior de la célula, hasta que impacta en el sustrato fotovoltaico, o bien son absorbidos por otras superficies. Cada vez que el rayo incide en una superficie, parte de él es reflejado, absorbido y/o refractado de acuerdo a leyes precisas bien conocidas. El estudio agregado del comportamiento de un gran número de rayos permite calcular muchas de las propiedades de interés en una célula fotovoltaica.

El software desarrollado permitirá analizar cuáles son los parámetros y condiciones que más influyen en la eficiencia de las células fotovoltaicas y proponer un modelo que permita optimizar la conversión de la energía solar, recibida por los fotoreceptores de una placa solar, en energía eléctrica.

Para analizar los parámetros que influyen de modo predominante en la absorción y conversión de energía solar en energía eléctrica, se ha usado como referencia una pieza de software desarrollada originalmente por la compañía australiana PV Lighthose Pty. Ltd. que permite modificar numerosos parámetros que puedan afectar a la eficiencia de la célula fotovoltaica. El software incluye varias muestras de radiación solar recogidas en distintos lugares y en instantes de tiempo concretos, con distintas condiciones climatológicas, que permiten realizar diferentes simulaciones y obtener resultados diversos que poder analizar una vez concluídas dichas simulaciones.

Tomando este software como partida, se ha desarrollado un paquete de código MATLAB de prestaciones similares. Los objetivos de diseño de este software han sido, (i) fiabilidad de resultados, para loque se han tratado de replicar los resultados del software arriba mencionado ante escenarios de simulación idénticos, (ii) flexibilidad para permitir adaptar el software a una variedad de geometrías y características ópticas de diferentes celdas, y (iii) modularidad, de modo que el software sea fácilmente integrable con paquetes de optimización numérica de MATLAB.

1.2 Justificación del Proyecto

La industria de las energías renovables va ganando en importancia año tras año. Concretamente en España, son comunes las energías eólica, hidroeléctrica y solar. Teniendo en cuenta que son fuentes de energía infinitas y limpias, principal diferencia con respecto a las fuentes de energía convencionales, las energías renovables parecen tener un importante papel en el futuro de la producción y el abastecimiento de la electricidad. Sin embargo, estas nuevas tecnologías siguen sin ser tan eficientes como sus predecesoras y es necesario seguir investigando cómo maximizar la eficiencia de la producción de energía.

Se estudiarán los factores que intervienen en mayor medida en la eficiencia de la conversión solar-eléctrica, especialmente los ángulos de incidencia de los rayos solares en los receptores de las células fotovoltaicas.

1.3 Introducción histórica

La primera noticia que se tiene acerca de la electricidad tiene lugar en la antigua Grecia. Hacia el 600 a.c., el filósofo griego Tales de Mileto (624 a.c. - 546 a.c), se dio cuenta de que, si frotaba una muestra de ámbar con lana o piel, la muestra atraía pequeños objetos de hierro. Estaba observando el efecto estático de la electricidad. Tales de Mileto no dio una explicación científica al fenómeno que estaba observando, si no que dotó al ámbar de alma para justificar el comportamiento que estaba teniendo lugar. El nombre griego del ámbar es *elektron,* de donde surge el término actual de electrón. No fue hasta unos 500 años más tarde cuando Tito Lucrecio Caro (99 a.c. - 55 a.c.), también filósofo griego y defensor de las teorías de Epicuro (341 a.c. - 246 a.c.), a su vez seguidor de Demócrito (460 a.c. - 370 a.c.), dijo que toda la naturaleza estaba compuesta por átomos. Esta teoría da un aire más científico al fenómeno del electromagnetismo, desechando las hipótesis de Tales de Mileto sobre el alma [1,2].

Durante más de mil años, no hubo apenas avances científicos en el campo del electromagnetismo. En el siglo XVI, el científico inglés William Gilbert (1544 – 1603), distingue por primera vez entre electricidad y magnetismo, y clasifica los materiales en eléctricos y aneléctricos [1,2].

Alrededor de medio siglo más tarde, el físico alemán Otto von Guericke (1602 – 1686), realizó una serie de experimentos en torno al electromagnetismo. Entre estos experimentos destaca la creación de una esfera de azufre ensartada en una varilla, con sus extremos libres de manera que pudiese girar con facilidad. Guericke había inventado la primera máquina eléctrica de la historia [1,2].

Más tarde la esfera de azufre se sustituyó por grandes cilindros de vidrio o esferas apoyadas en marcos de madera, en contacto con materiales de cuero u otros elementos que hiciesen roce y produjeran electricidad. Posteriormente, la carga del cilindro o cuerpo esférico se transfería a una pieza de metal más grande [1,2].

En 1733, el científico francés Charles François de Cisternay Du Fay (1698 – 1739), observó, utilizando una lámina de oro, que ésta era repelida siempre por una barra de vidrio electrificada. A partir de sus experimentos, Du Fay determinó que existían dos tipos de electricidad, que él denominó carga vítrea (cuando la lámina de oro era repelida) y carga resinosa (cuando era atraída). Unos años más tarde, el estadounidense Benjamin Franklin (1706 – 1790), las rebautizaría como carga positiva y carga negativa [3].

En 1745, el físico holandés Pieter van Musschenbroek (1692 – 1761), comprobó si en una botella de agua se conservaban cargas eléctricas. El exterior de la botella estaba recubierto por papel de aluminio. Esta disposición se conoce aún hoy día como *botella de Leyden*, en honor a la universidad en la que trabajaba van Musschenboek. Para comprobar si había electricidad almacenada dentro de la botella, van Musschenboek tocó tanto dentro como fuera del tarro corroborando que así era. La botella de Leyden encontraría muchas aplicaciones en los años venideros y supondrían la base de los actuales condensadores eléctricos. Sir William Watson (1715 – 1787), desarrolló en 1747 la botella de Leyden añadiéndole una placa de metal, comprobando que aumentaba la descarga eléctrica y demostrando que ésta era una corriente eléctrica [1,2,4].

En 1752, el anteriormente mencionado Benjamin Franklin, demuestra la naturaleza eléctrica de los rayos y lanza la hipótesis de que la electricidad es un fluido existente en la materia y que su flujo se debe al exceso o defecto de éste en ella [4].

El ingeniero francés Charles-Augustin de Coulumb (1736 – 1806) fue el primero en establecer leyes cuantitativas sobre la electrostática. Siguiendo las hipótesis de Joseph Priestly (1732 – 1804), Coulumb, en 1785, confirma la naturaleza del inverso cuadrático de la fuerza eléctrica. Para ello, intentó una balanza de torsión para medir tangiblemente la fuerza de atracción o repulsión entre dos cargas eléctricas, en función de la distancia que las separa [1,2].

Luigi Galvani (1737 – 1798) fue un científico italiano que pasaba mucho tiempo diseccionando animales. En una de estas disecciones, con un ejemplar de rana, Galvani tocó accidentalmente una de las patas de la rana con un bisturí, el cual produjo una descarga y contrajo la pata. A raíz de este suceso, Galvani realizó una serie de experimentos en los que determinó que la contracción de la pata de la rana se debía a la estimulación por parte de una máquina eléctrica. El también científico Alessandro Volta (1745 – 1827) reprodujo los experimentos de Galvani y advirtió, como hiciera Galvani en su momento, que la presencia de electricidad en la contracción de la pata de la rana se producía cuando dos metales diferentes, cada uno tocando al otro, tocaban con sus otros extremos la pata de la rana. Tomando la hipótesis de que la electricidad emanaba de los alambres de metal y no del cuerpo de la rana, tomó dos tiras metálicas (de latón y zinc) con un cartón empapado de agua salada entre ellas. Para intensificar el efecto, lo repitió sesenta veces colocando unas encima de otras. De esta manera, Volta inventó la batería, que proporcionó carga eléctrica en grandes cantidades [2,4].

A finales de 1820, el francés André-Marie Ampère (1775 – 1836), después del descubrimiento por parte de Hans Christian Ørsted (1777 – 1851) de que un flujo de corriente eléctrica en un cable podía desviar una aguja de brújula magnética cercana, decidió estudiar a fondo este fenómeno. Ampère, reprodujo los experimentos de Østerd y determinó que, si la corriente eléctrica fluye en la misma dirección dentro de dos cables paralelos cercanos, dichos cables se atraen entre sí. Sin embargo, si las corrientes eléctricas de los cables fluyen en direcciones contrarias, los cables se repelen. De esta forma se produce atracción y repulsión magnética en total ausencia de imanes. Ampère denominó a este nuevo campo de la ciencia electromagnetismo [4].

En los siguientes años, se produjeron numerosos avances en el campo del electromagnetismo. Georg Simon Ohm (1789 – 1854), relacionó matemáticamente el voltaje aplicado a una resistencia eléctrica y la corriente eléctrica que la recorre. Joseph Henry (1797 – 1878) y Michael Faraday (1771 – 1867), descubrieron por separado la inducción eléctrica, base para la construcción de generadores y motores de inducción. En 1834, Heinrich Lenz (1804 – 1865), dedujo la ley que lleva su nombre; “una corriente eléctrica inducida fluye en una dirección tal que la corriente se opone al cambio que la indujo”. El descubrimiento de la inducción electromagnética revolucionó el uso de la energía eléctrica [2,4].

En 1845, Gustav Kirchhof (1824 – 1887), enuncia dos leyes que permiten calcular analíticamente la distribución de tensiones y de corrientes en redes eléctricas [4].

A finales del siglo XIX, la electricidad comenzó a introducirse en la vida cotidiana con la aplicación del motor eléctrico a algunos inventos de la época como el ascensor eléctrico. En 1880, Thomas Alva Edison (1847 – 1931), patentó la bombilla incandescente, una de las primeras aplicaciones de la electricidad a la vida moderna. Edison convirtió la electricidad en un bien comercial y dio los primeros pasos para la construcción de las primeras líneas eléctricas que abastecieran electricidad a sus clientes [2,4].

A partir de entonces, la electricidad se convirtió en un modelo de negocio diferente a cualquier otro, ya que se caracteriza por ser muy difícil y costosa de almacenar, por lo que es necesario consumirla en el momento en que se genera [1,2,4].

Desde que Thomas Edison, con la inversión inicial de J.P. Morgan, creara, en 1882, la primera central de generación de energía eléctrica en Manhattan, Nueva York, el consumo de electricidad ha crecido exponencialmente en todo el mundo. En el siguiente gráfico, se muestra el consumo mundial de energía eléctrica, en TW/h, desde la segunda guerra mundial hasta la actualidad [1,2,4]. (Fig.1).

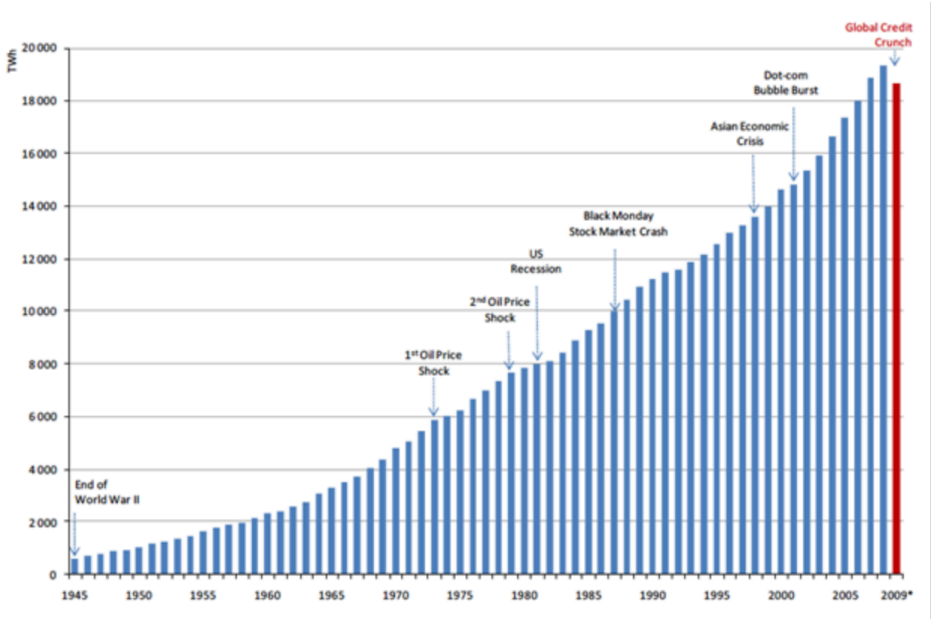

Figura 1. Demanda mundial de electricidad. Fuente: IEA Databases anda analysis.

La electricidad se ha vuelto indispensable para la sociedad mundial. A raíz de este consumo y demanda exponencial, a lo largo del siglo XX se han ido desarrollando nuevas formas de generar energía eléctrica. Se discuten en el punto que viene a continuación [1,2,4].

2 GENERACIÓN DE ENERGÍA ELÉCTRICA

2.1 Industria eléctrica a nivel mundial

Como se ha visto en el punto 1.3, la electricidad no es un bien que pueda extraerse en grandes cantidades de la naturaleza, si no que es necesario generarla. Además, la electricidad tiene un inconveniente notable, no es posible almacenarla en grandes cantidades, por lo que hay que consumir lo que se genere inmediatamente. La generación de electricidad consiste básicamente en la conversión de otros tipos de energías, que sí son abundantes en la naturaleza, en energía eléctrica. Esta conversión se produce de distintos modos, y la utilización de todos ellos, los más eficientes, rentables y no nocivos para el medio ambiente constituyen la industria energética [5].

Como se puede comprobar en la figura adjunta (Fig. 2), las fuentes de generación de energía más utilizadas en el mundo son: Combustibles fósibles, energía nuclear y energías renovables (en este trabajo se tratará la energía hidráulica como energía renovable) [4,5].

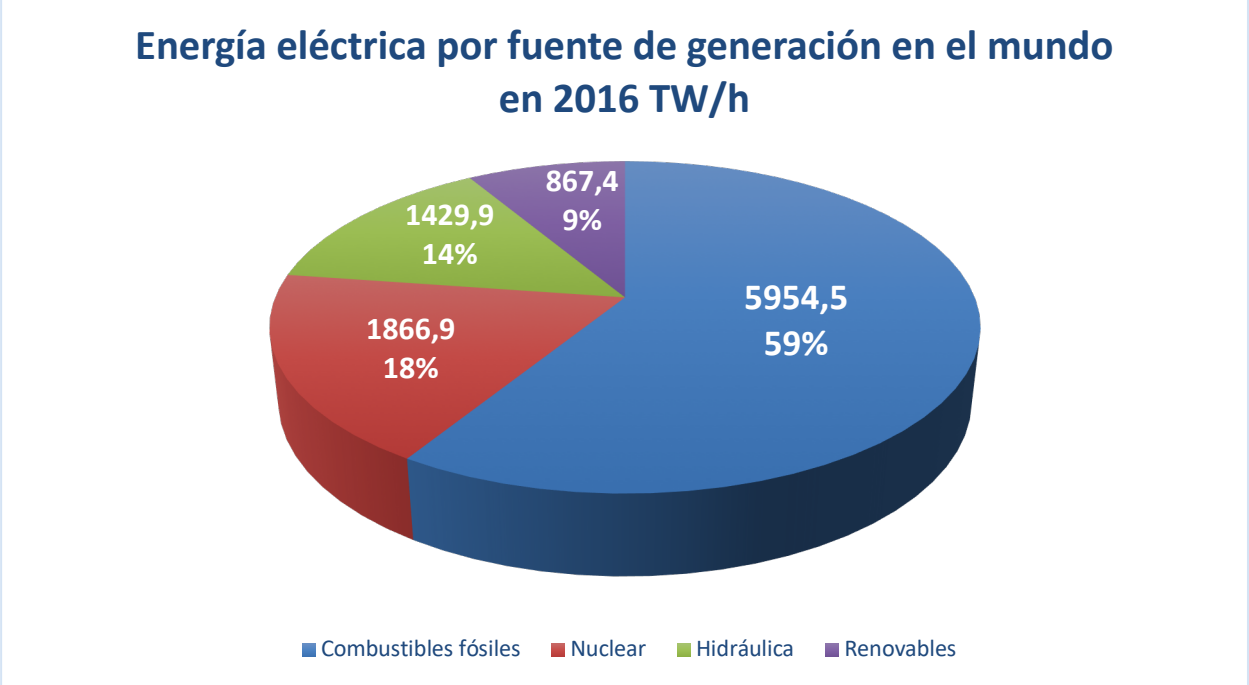

Figura 2. Energía eléctrica por fuente de generación en el mundo en 2016 TW/h.
Fuente: IEA - Monthly electricity statistics, pp. 5, Nov. 2017.

2.1.1 Centrales Nucleares

La centrales nucleares son plantas industriales en las que se genera energía eléctrica en grandes cantidades, del orden de los MW/h. La energía nuclear se ha ido desarrollando sobre todo a partir de la segunda mitad del siglo XX. La primera central nuclear de la historia fue construida en Óbninsk, actual Rusia, en 1954. Desde entonces, la energía nuclear ha seguido creciendo hasta nuestros días, convirtiéndose en un importante pilar en la generación de energía eléctrica. Los países con más centrales nucleares del mundo son Estados Unidos, con 104 centrales nucleares, Francia con 58 y Japón con 44 [10], aunque, según la IAEA (International Atomic Energy Agency), hay 449 reactores nucleares en funcionamiento en 31 países diferentes [6,8].

A pesar de ser una energía barata y limpia, como se describe a continuación, la energía nuclear no ha estado exenta de pólemica debido a varios accidentes nucleares ocurridos en distintas partes del mundo, entre los que destacan el de la planta nuclear de Three Mile Island, Estados Unidos, en 1979; Chernobyl, Ucrania, en 1986 y Fukushima, Japón, en 2011[7].

Una central nuclear es básicamente una central térmica en la que la fuente de calor es un reactor nuclear. El calor generador en el reactor nuclear produce vapor que se utiliza para impulsar una turbina de vapor, que a su vez está conectada a un generador eléctrico que produce electricidad [6].

Existen varios tipos de centrales nucleares siendo el reactor de agua a presión, conocido como PWR (Pressurized Water Reactor) por sus siglas en inglés, el tipo más común de ellos. En la Fig.3 se presenta un esquema general de una planta de PWR con los distintos tipos de sistemas que la componen [9].

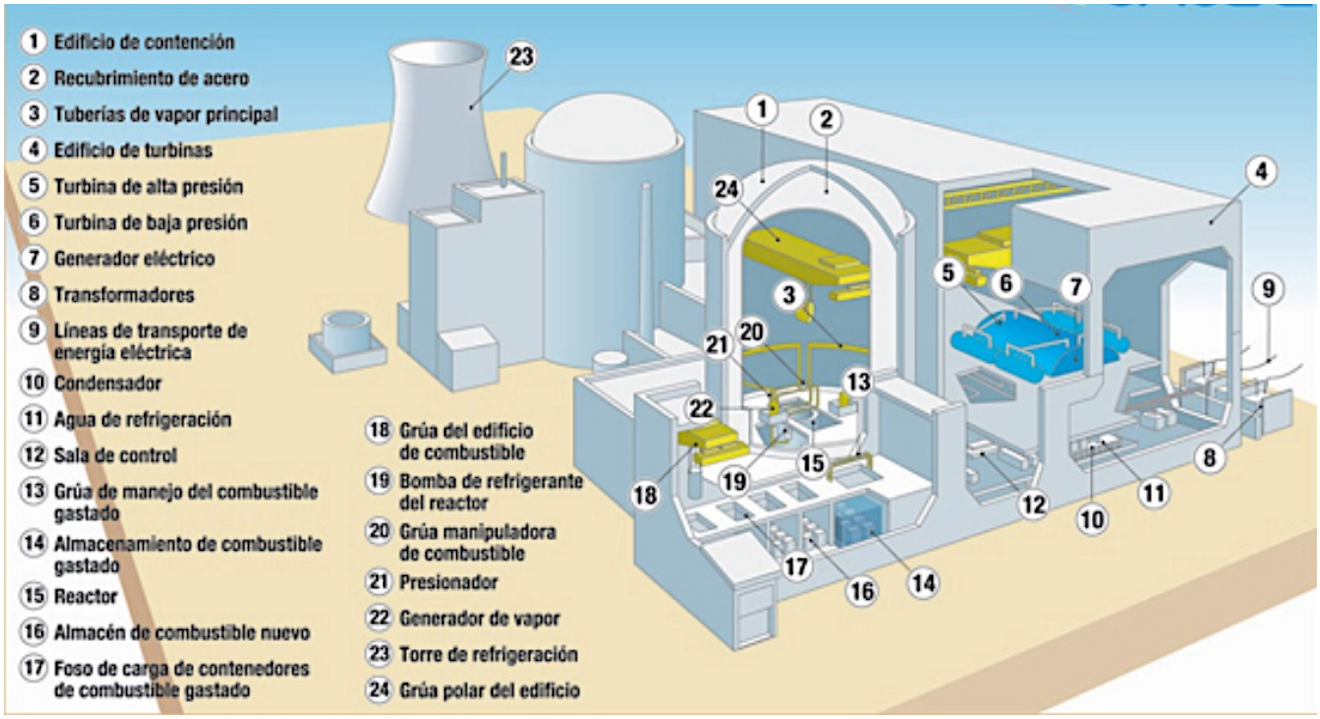

Figura 3. Componentes de una central nuclear.
Fuente: UNESA (http://www.unesa.es/sector-electrico/funcionamiento-de-las-centrales-electricas/1349-central-nuclear)

El edificio de contención (1) contiene los principales sistemas del circuito primario. El fluido refrigerante, en este tipo de central agua, es calentado con el calor producido en la fisión de los núcleos de Uranio ocurridos en el reactor. El refrigerante se mantiene en estado líquido debido a la alta presión a la que es sometido y es conducido hacia los generadores de vapor. En los generadores de vapor, el agua del circuito secundario es convertida en vapor, el cual es conducido por las tuberías de vapor principal (3) del edificio de contención hasta el edificio de turbinas (4). En este edificio, el vapor pone en movimiento las turbinas (5 y 6) que convierten este movimiento en energía eléctrica gracias a la acción de un generador eléctrico (7). Una vez que el vapor sale de las turbinas, vuelve a estado líquido en el condensador (10). El agua vuelve a ponerse en su estado inicial a través de una o varias torres de refrigeración (23) que la devuelven al lugar de donde se extrajo (ríos o mares) [6,7].

Un edificio indispensable de las centrales nucleares son los almacenes de combustible. En el almacén de combustible gastado (14) se permite la pérdida gradual de actividad de combustible ya usado, que posteriormente se descontamina y se lleva fuera de la central. La energía eléctrica producida en el proceso de fisión nuclear pasa por un transformador eléctrico (8) que la traslada a la red de línea eléctrica para su distribución posterior [8,9].

El segundo tipo de central más frecuente es el reactor de agua en ebullición, BWR (Boiling Water Reactor). Su principal diferencia con el PWR es que el BWR emplea sólo un circuito en el que el refrigerante (agua) se convierte directamente en vapor por el calor producido en la fisión, mueve las turbinas y se condensa actuando a su vez como refrigerante. Tanto el PWR como el BWR utilizan Uranio enriquecido como combustible ^{235}U. En la Fig.4 se puede ver el esquema de un reactor tipo BWR [6].

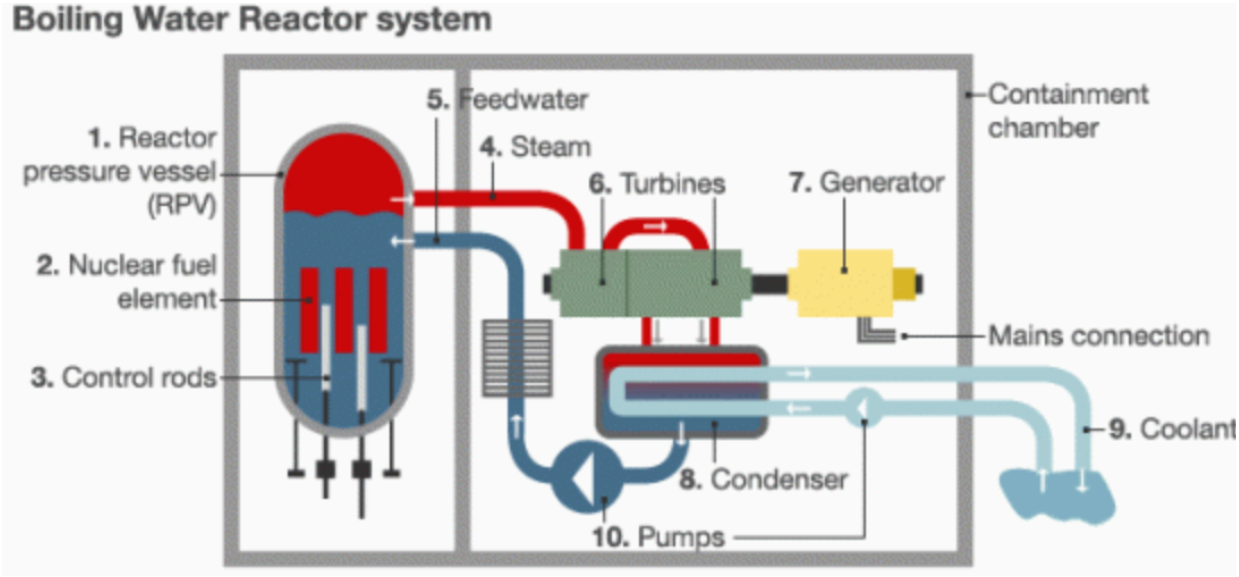

Figura 4. Componentes de un reactor de agua pesada a presión.
Fuente: http://desastrenuclearfukushima.blogspot.com.es/2012/06/1funcionamiento-de-una-central-nuclear_03.html

Los otros tipos de central nuclear, mucho menos frecuentes que las anteriores, son: reactor de agua pesada a presión, PHWR (Pressurized Heavy Water Reactor); reactor avanzado refrigerado por gas, AGR (Advanced gas-cooled reactor), que sólo existe en Reino Unido; reactor de condensador de alta potencia, RBMK de sus siglas en ruso y, reactor reproductor, FBR (Fast Breeder Reactor).

En el aspecto económico, las centrales nucleares tienen unos costes de inversión muy elevados, actualmente unos 4.000 millones de euros por Gigavatio de potencia instalado, pero tienen unos costes de explotación moderadamente bajos ya que, aunque el combustible en sí es caro, no se precisan grandes cantidades del mismo para su explotación [6]. En cuanto a la contaminación ambiental, no existen emisiones de gases de efecto invernadero por lo que la energía nuclear puede considerarse una energía limpia. Sin embargo, sí quedan residuos radiactivos que deben ser tratados convenientemente después de su uso. Además, el Uranio enriquecido es un recurso natural agotable al ritmo de explotación actual, por lo que no es una técnica de conversión de energía que pueda usarse eternamente [8].

Por otro lado, el rendimiento de la conversión de las centrales nucleares se sitúa en torno al 30%-35% calculando con la ecuación (1) [8].

$$\frac{P_{eléctrica}}{P_{recurso\ primario}} \quad (1)$$

2.1.2 Combustibles fósiles

Como se ve en el gráfico de la Fig.2, la generación de energía eléctrica por medio de combustibles fósiles sigue siendo el método mayoritario de obtención de la misma. Existen diversos tipos de combustibles para la generación de energía eléctrica a partir de combustibles fósiles, así como varios procesos para ello, que se explican a continuación. Las dos principales desventajas que presenta esta forma de generación son, por un lado, el hecho de depender de unos combustibles o recursos limitados en la Tierra, y por otro lado el elevado grado de emisiones contaminantes que conlleva utilizarlos. Los combustibles fósiles más utilizados en la industria de la energía eléctrica son el carbón, el gas natural y el combustóleo [8,9,11].

2.1.2.1 Central de ciclo combinado

El ciclo combinado es un método de generación de energía eléctrica en el que se utilizan dos procesos termodinámicos, uno de ellos, ciclo de Brayton, mediante la combustión de un gas, generalmente gas natural, y el otro, ciclo de Rankine, el convencional agua/turbina de vapor. La Fig. 5 – muestra un esquema general de una central de ciclo combinado [8,11].

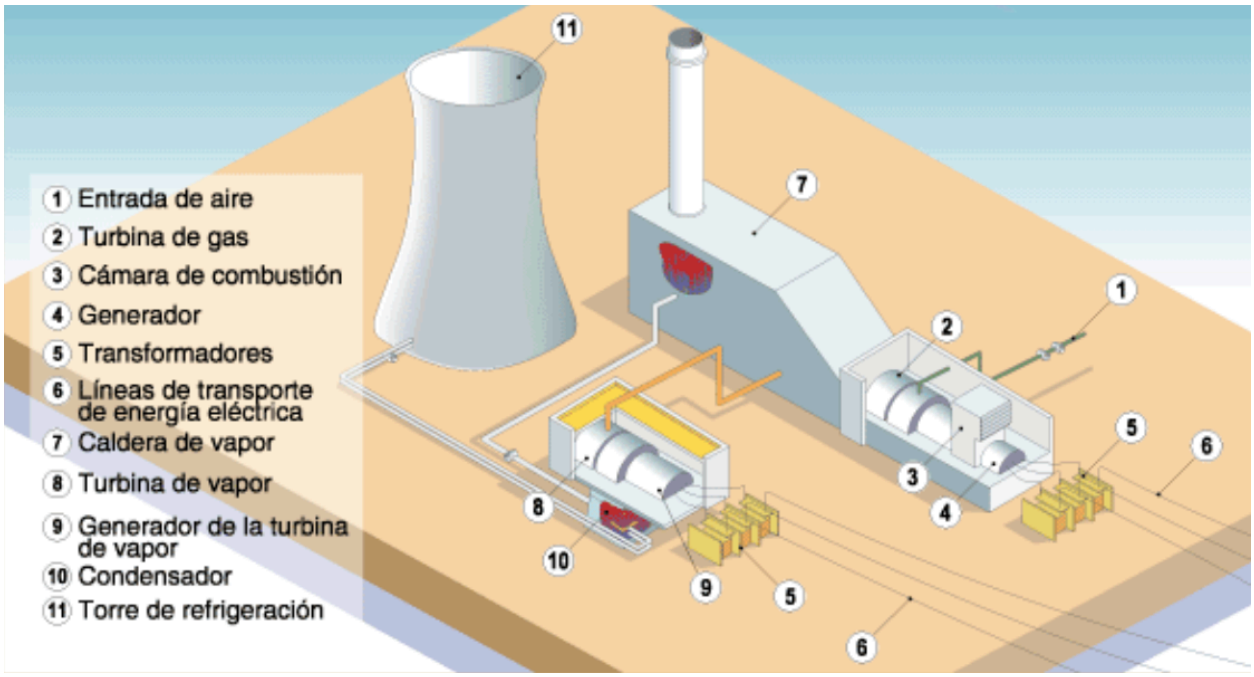

Figura 5. Componentes de una central de ciclo combinado.
Fuente: UNESA (http://www.unesa.es/sector-electrico/funcionamiento-de-las-centrales-electricas/1343-central-ciclo-combinado)

En la turbina de gas (2) se somete el aire a elevadas presiones mediante un compresor para mezclarlos más tarde con el gas en la cámara de combustión (3). En este punto se produce la combustión de la mezcla en unas condiciones de presión y temperaturas óptimas para maximizar el rendimiento de ésta [11].

Los gases de combustión se dirigen a la turbina de gas moviendo un eje de rotación con la energía mecánica obtenida que a su vez acciona un generador eléctrico y transforma esta energía mecánica en electricidad. Esta electricidad pasa por un transformador (5) que la convierte de media tensión a alta tensión para evitar pérdidas y la envía a la red de distribución (6) [11].

El rendimiento en la turbina aumenta con la temperatura, que llegan a alcanzar los 1.300 ºC a la entrada y aún conservan unos 600ºC en la última etapa. Con esa temperatura de salida aún puede aprovecharse la energía mecánica que conservan, por lo que se conducen los gases a la caldera de vapor (7) donde se convierte agua en vapor. [11] Desde ese momento se da el convencional proceso agua/turbina de vapor; se dirige el vapor de agua a la turbina de vapor, que mediante el movimiento de los álabes acciona el eje que mueve el generador eléctrico. De nuevo la energía eléctrica generada es de media tensión y se transforma a alta tensión para evitar pérdidas de transporte antes de pasarla a la red de distribución [12].

En las centrales modernas se unen generalmente ambos ciclos en el mismo generador eléctrico para reducir costes de infraestructura [12].

El coste de inversión de las centrales de ciclo combinados es moderado, alrededor de los 260.000€ por una central de potencia neta de 400MW [12]. Produce emisiones moderadas de CO_2 y reducidos de SO_2 y NO_x como consecuencia de la combustión en el ciclo de Brayton. Su rendimiento medio atendiendo a la expresión de la ecuación (1) es en torno al 50%-60% de conversión. El fin de los recursos de gas natural en el mundo se estima en unos 80 años desde la actualidad [11,13].

2.1.2.2 Central de gasificación integrada con ciclo combinado

Estas centrales utilizan la gasificación del combustible primitivo, generalmente carbón, para la generación de electricidad. En un principio se gasifica la materia prima, se expande dicho gas en una turbina y se aprovecha posteriormente el calor residual para alimentar una turbina de vapor. La Fig.6 muestra un esquema general de una central de gasificación integrada con ciclo combinado.

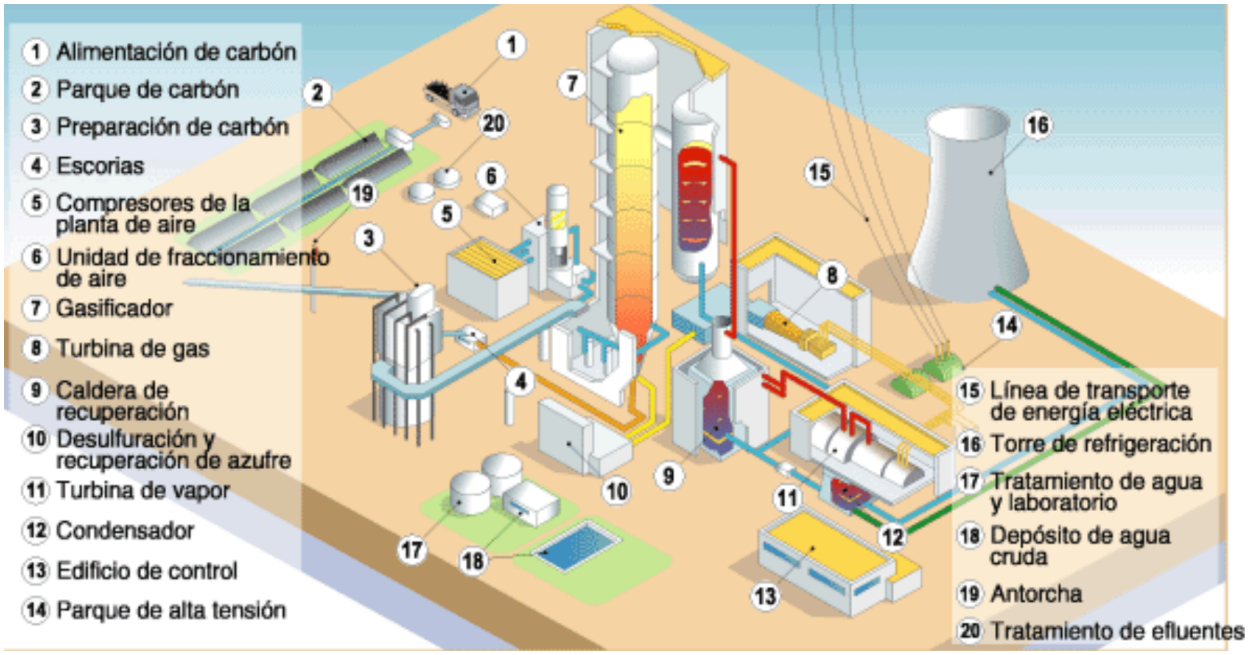

Figura 6. Componentes de una central de gasificación.
Fuente: UNESA (http://www.unesa.es/sector-electrico/funcionamiento-de-las-centrales-electricas/1346-central-gasificacion)

En primer lugar, se pulveriza el carbón, se separa de las escorias y se seca mediante nitrógeno. Seguidamente, el combustible pulverizado se enfría para extraer oxígeno que se utilizará en la gasificación del carbón. En el gasificador (7) se introduce el combustible junto con el oxígeno extraído previamente y vapor de agua, formándose un gas sintético a muy alta temperatura.

Antes de ser quemado, el gas se somete a una desulfuración (10) y se conduce a un grupo de gas compuesto por un compresor y una toma de aire exterior que adapta las condiciones para una combustión conveniente. Los gases de combustión accionan la turbina que mueve el generador que produce electricidad. La electricidad se dirige los transformadores que adapta su tensión e intensidad a la de la red de transporte.

El calor residual de los gases de combustión se aprovecha en la caldera de recuperación (9) para convertir en vapor de agua el contenido del depósito (18). El vapor se conduce a una turbina de vapor que acciona el generador eléctrico, que pasa la energía eléctrica a los transformadores y de ahí a la red de distribución.

El coste de inversión en una planta de gasificación con ciclo combinado es moderado y tiene unos precios de explotación medios según el precio del combustible, 90 dolares por tonelada de carbón en 2016 [11,14]. El uso de estas centrales supone unos altos niveles de emisión de CO_2, SO_2 y NO_x a la atmósfera lo que provoca altos niveles de contaminación. El rendimiento de estas

centrales se sitúa entre el 30% y el 45% de conversión según la ecuación (1). Las reservas de carbón en la Tierra se estiman duraderas por unos 200 años más [14]

2.1.2.3 Otras centrales

Otras centrales generadoras dignas de mención son:

- Central de cogeneración mediante biomasa: Tienen un funcionamiento similar a las centrales térmicas convencionales, pero usan como combustibles principales residuos forestales, cultivos de plantas energéticas y residuos agrícolas. En cuanto a las emisiones de CO_2 se estima que serán prácticamente iguales que las que se producirían en un proceso de descomposición natural [11].
- Central de residuos sólidos urbanos: También se diferencian de las centrales térmicas convencionales en el combustible utilizado, aparte de adaptar la cámara de combustión para este tipo de residuos. En este caso se utilizan residuos urbanos y es una técnica muy utilizada en Europa [11].
- Central térmica convencional de carbón: Funcionamiento similar al explicado en el punto 2.1.2.1 utilizando tan sólo un ciclo termodinámico para la generación de energía eléctrica [11].

2.2 Industria eléctrica en España

La energía eléctrica en España procede de diversos orígenes y, aunque los combustibles fósiles siguen siendo la principal fuente de generación de electricidad como se ve en la Fig. 7, el peso que de las renovables en esta industria va en aumento, como también se aprecia en la Fig. 8 [13, 19].

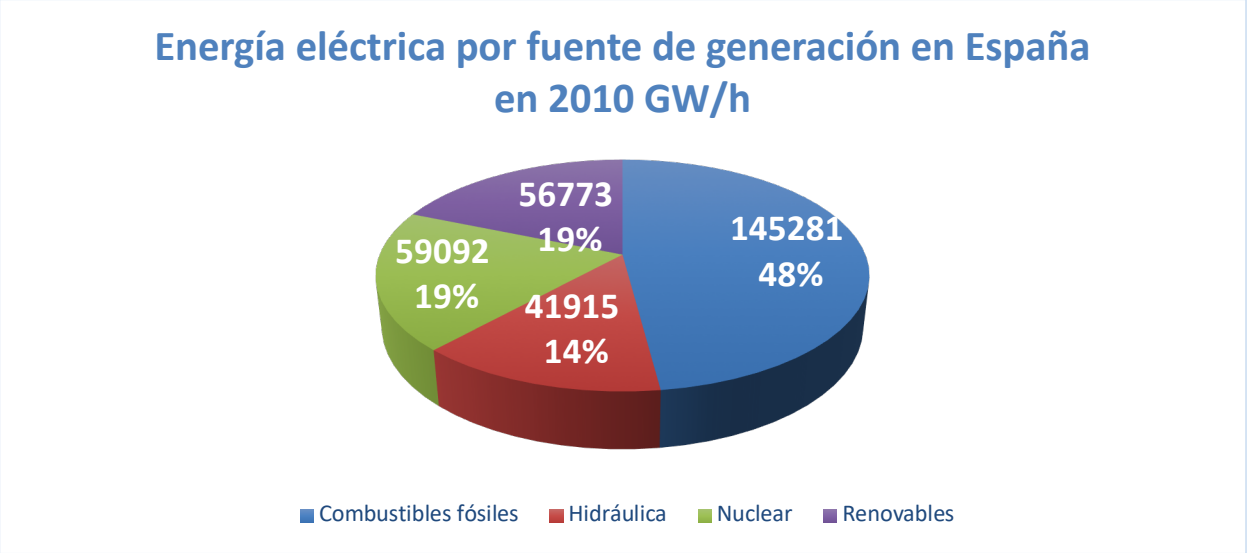

Figura 7. Energía eléctrica por fuente de generación en el mundo en 2010 GW/h.
Fuente: Sedigas (https://www.sedigas.es/informeanual/2010/2.3_ConsumoEnergiaFinal.htm)

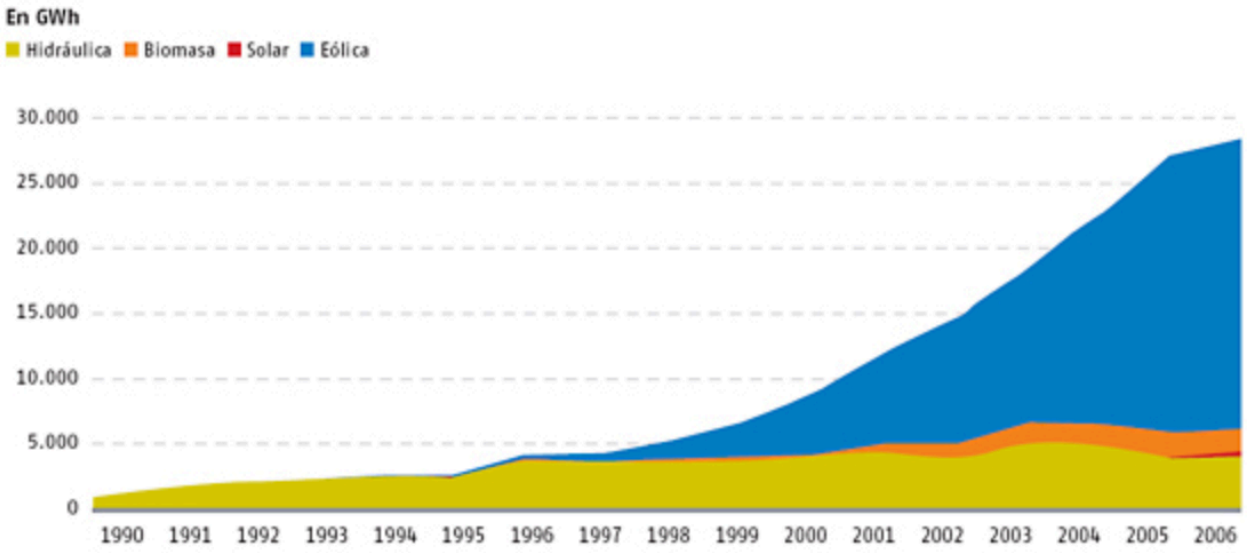

Figura 8. Evolución de la generación de orgien renovables 1990-2006.
Fuentes: CNE y AEE (https://news.soliclima.com/divulgacion/energia-eolica/el-sector-de-la-energia-eolica-en-2006)

2.2.1 Energía nuclear en España

La energía nuclear en España comenzó a desarrollarse a mitad del siglo XX, tras la segunda guerra mundial. En total se han construido en España 10 reactores en 7 emplazamientos diferentes. Tres de estos reactores nucleares se encuentran en desmantelamiento, mientras que los otros siete continúan operando con una potencia total instalada de 7.728 MWe. Los datos más relevantes de los 10 reactores nucleares de España vienen recogidos en la siguiente Tabla 1 [13, 15,16].

Central	Emplazamiento	Propietarios	Potencia Eléctrica (MW)	Tipo	Año de entrada en servicio	Año de fin de licencia/fin de actividad
Sta. María Garoña	Valle de Tobalina (Burgos)	Iberdrola Generación S.A. (50%), Endesa Generación S.A. (50%)	466,00	BWR	1971	2012
Almaraz I	Almaraz (Cáceres)	Iberdrola Generación S.A. (52,7%), Endesa Generación S.A. (36%), Gas Natural S.A. (11,3%)	1.035,30	PWR	1981	2021
Almaraz II	Almaraz (Cáceres)	Iberdrola Generación S.A. (52,7%), Endesa Generación S.A. (36%), Gas Natural S.A. (11,3%)	1.045,00	PWR	1983	2023
Ascó I	Ascó (Tarragona)	Endesa Generación S.A. (100%)	1.032,50	PWR	1983	2023
Ascó II	Ascó (Tarragona)	Endesa Generación S.A. (85%), Iberdrola Generación S.A. (15%)	1.027,21	PWR	1985	2025
Cofrentes	Cofrentes (Valencia)	Iberdrola Generación S.A. (100%)	1.092.02	BWR	1984	2024
Vandellós I	Vandellós (Tarragona)	-	480,00	GCR	1972	1989
Vandellós II	Vandellós (Tarragona)	Endesa Generación S.A. (72%), Iberdrola Generación S.A. (28%)	1.087,14	PWR	1987	2027
Trillo	Trillo (Guadalajara)	Iberdrola Generación S.A. (48%), Gas Natural S.A. (34,5%), Hidroeléctrica Cantábrico (15,5%), Nuclenor (2%)	1.066,00	PWR	1988	2028
José Cabrera-Zorita	Almocid de Zorita (Guadalajara)	-	160,00	PWR	1969	2006

Tabla 1. Relación de centrales nucleares en España. [17]
Fuente: Ministerio de energía, turismo y agenda digital
(http://www.minetad.gob.es/energia/nuclear/Centrales/Espana/Paginas/CentralesEspana.aspx)

PWR: Reactor de agua a presión.
BWR: Reactor de agua en ebullición.
GCR: Reactor refrigerado por gas.

La ventaja principal de las centrales nucleares es que están todo el tiempo en funcionamiento, excepto una parada al año que se realiza para revisar toda la instalación. Tras el cierre de una central nuclear, comienza el desmantelamiento de la misma que se alarga por unos 13-16 años en el tiempo para dejar toda la zona limpia de radiactividad. Vandellós I está en periodo de latencia, mientras que José Cabrera-Zorita y Sta. María Garoña están en desmantelamiento [16,20].

Como se dijo en el **punto 2.1.1**, el combustible empleado en las centrales nucleares es Uranio enriquecido. Son necesarias unas 150 toneladas anuales de Uranio enriquecido para el abastecimiento de las centrales nucleares españolas y es 100% importado del exterior, lo que supone un gasto extra para la explotación del recurso con el fin de generar energía eléctrica [16,20].

2.2.2 Centrales térmicas en España

En España hay aproximadamente 200 centrales térmicas, con una potencia total instalada de más de 27.000 MW, es decir, 140 MW de media por central. De todas las centrales térmicas de España, sólo 6 tienen una potencia instalada de más de 1.000 MW:

- Puentes de García Rodríguez (La Coruña): es la central térmica más grande de España con 1.468 MW de potencia instalada de ciclo convencional y 800 MW de ciclo combinado. El combustible utilizado es carbón.
- Compostilla (León): 1.312 MW de potencia instalada. Consta de 4 grupos térmicos, con el 2º grupo con limitaciones por no tener desulfurador (ver **figura -**), y utiliza carbón extraído de una cuenca minera local como combustible.
- Litoral de Almería (Carboneras, Almería): 1.100 MW de potencia instalada, 2 grupos térmicos. Emplea carbón importado como combustible.
- Central térmica de Castellón (Castellón de la Plana): 1.083 MW de potencia instalada utiliza fuel-oil como combustible.
- Central térmica de Andorra (Andorra, Teruel): 1.050 MW, emplea carbón procedente de las minas aragonesas de la zona. Tiene programado su cierre en 2020.
- San Adrián (Barcelona): 1.050 MW de potencia instalada con fuel y gas natural como combustibles.

En el caso de los combustibles fósiles utilizados en las centrales térmicas españolas, prácticamente el 100% del gas natural y el petróleo es importado de otros países, así como la mitad del carbón y gasóleo empleados en la generación de electricidad [20, 21, 22, 23].

2.2.3 Distribución de la energía eléctrica

Cada país tiene sus propias normas para el transporte de la energía eléctrica. En España se utiliza la norma UNE-EN 60909-0 que establece los tipos de cable, las tensiones, intensidades y transformaciones que deben hacerse según el tipo de instalación que quiera hacerse [13].

El sistema de distribución de energía eléctrica se encarga del transporte de la energía eléctrica generada desde las centrales de generación hasta el cliente o consumidor final de dicha energía. La electricidad, como se vio en los **puntos 2.2.1 y 2.2.2** una vez se ha generado la electricidad en la central, pasa por un transformador que adecúa la tensión e intensidad a la subestación eléctrica contigua Fig. 9. Estas subestaciones eléctricas trabajan a muy alta tensión (MAT). En España, se entiende por MAT valores de tensión que van desde los 220 kV hasta los 400 kV aproximadamente [17].

Figura 9. Subestación de Santa Llogaia, Cataluña.
Fuente: Agencia EFE

Los otros tipos de redes de distribución según la tensión eléctrica que soportan, en el sistema español son:

- Baja Tensión (BT): tensiones inferiores a 1 kV.
- Media Tensión (MT): tensiones comprendidas entre 1 kV y 30 kV.
- Alta Tensión (AT): 45 kV, 55 kV, 66 kV, 110 kV, 132 kV (la más frecuente).

Las líneas de distribución se dividen en líneas aéreas, subterráneas y submarinas y pueden diseñarse para soportar tanto MAT como AT, MT y BT. En los lugares urbanos existen los

conocidos como centros de transformación (CT) Fig. 10 - que adecúan la electricidad a la tensión e intensidad que marca la norma española para su uso final por los clientes en hogares, establecimientos, etc [11,20].

Figura 10. Centro de transformación en Isona, Lérida.
Fuente: lainformacion.com

2.3 Energías renovables a nivel mundial: Energía solar

La industria de las energías renovables ha ido creciendo exponencialmente desde la década de 1980 hasta nuestros días, como se puede apreciar en la gráfica de la Fig. 11. El crecimiento del consumo energético mundial y el compromiso con el respeto al medio ambiente han ayudado mucho al fomento de la innovación de las energías limpias. Aunque hay numerosas tecnologías de energías renovables, las más importantes son la energía eólica, la energía hidráulica y, la que más incumbe a este trabajo, la energía solar [24].

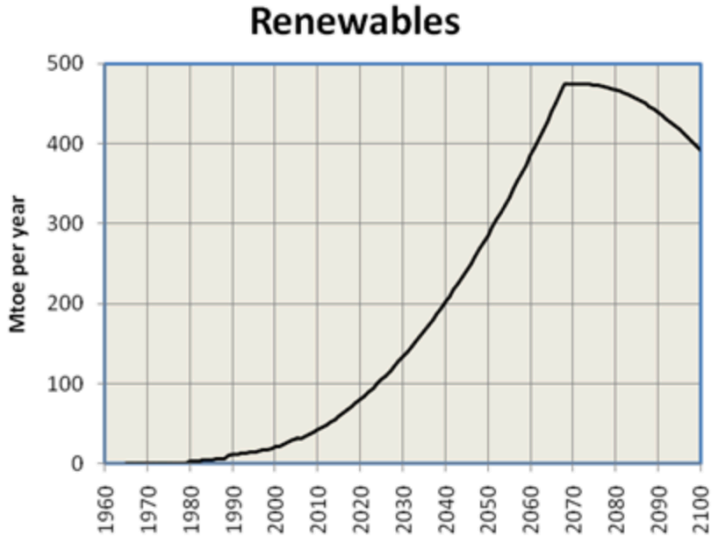

Figura 11. Evolución histórica de las energías renovables a nivel mundial.
Fuente: (http://www.paulchefurka.ca/WEAP/WEAP.html)

2.3.1 Energía eólica

La energía eólica consiste en el aprovechamiento de la energía cinética del viento para su conversión en energía eléctrica. La conversión se realiza a través de unas turbinas eólicas que, accionadas por el viento, alimentan mecánicamente un generador eléctrico que produce la electricidad. Estas turbinas eólicas, conocidas como aerogeneradores eléctricos, se concentran en centrales o parques eólicos como el de la Fig.12 [11, 14, 25].

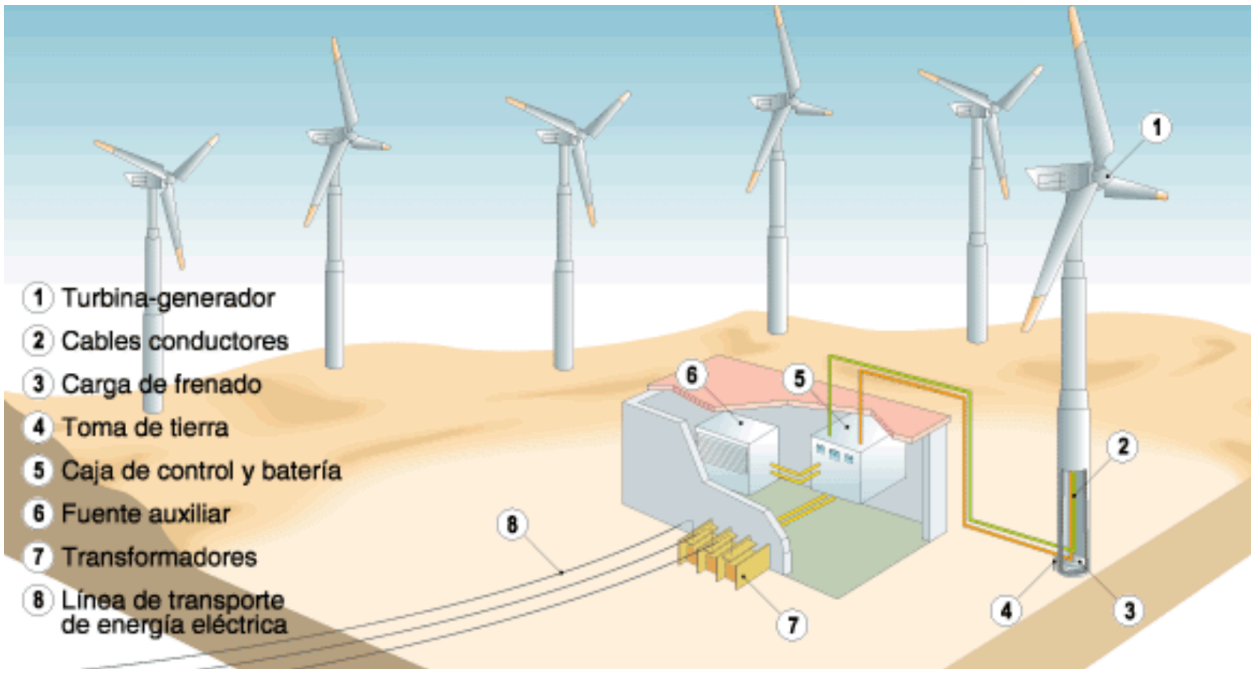

Figura 12. Funcionamiento de una central eólica.
Fuente: UNESA (http://www.unesa.es/sector-electrico/funcionamiento-de-las-centrales-electricas/1344-central-eolica)

Atendiendo a la Fig. 12, el viento mueve la turbina (1) generando electricidad que pasa por los cables conductores (2) y se conduce a los transformadores (7) para adaptarla a la red eléctrica de distribución (8) [11].

Los costes de inversión para los parques eólicos son moderados, alrededor de 1 millón de euros por una central de 23,4 MW de potencia instalada). Los costes de mantenimiento se estiman por los 17.500€ por MW y tienen una vida útil de unos 20 años [14].

Algunos países como Reino Unido, Holanda o Dinamarca han construido parques eólicos marítimos, evitando contaminación visual y ahorrando espacio en tierra firme, aunque la construcción de estos parques marítimos resulta más cara que los convencionales [14].

En cuanto al rendimiento en la conversión de la potencia del recurso primario, viento, en potencia eléctrica, se estima en torno al 20%-30% de conversión atendiendo a la ecuación (1) [11,14].

2.3.2 Energía hidráulica

La energía hidráulica utiliza la energía potencial gravitatoria del agua para generar electricidad. Se emplea en centrales hidroeléctricas como la de la Fig. 13 [11,26].

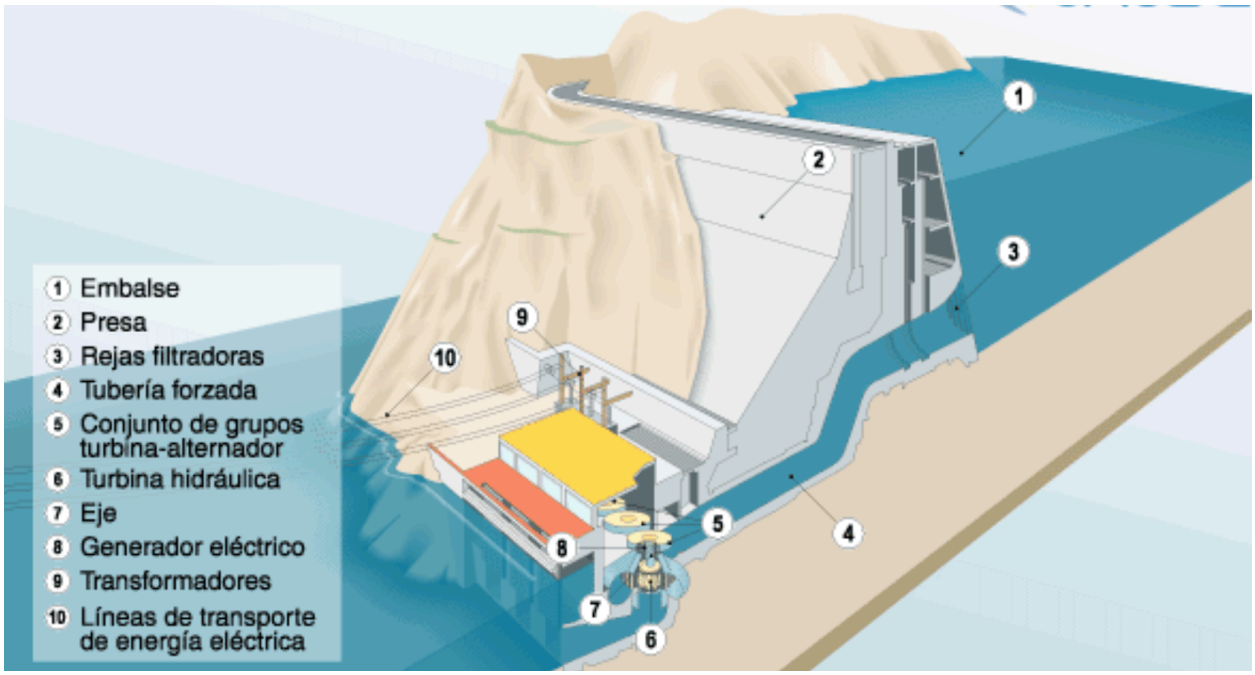

Figura 13. Componentes y funcionamiento de una central hidroeléctrica.
Fuente: UNESA (http://www.unesa.es/sector-electrico/funcionamiento-de-las-centrales-electricas/1347-central-hidroelectrica)

La infraestructura que se ve en la figura corresponde a un embalse en el lecho de un río. El agua se acumula artificialmente en el embalse lo que hace que adquiera energía potencial. Las rejas filtradoras (3) controlan el caudal de agua que se utilizará para la generación de electricidad. Aguas abajo del embalse se situa un conjunto de turbinas generadoras (5) que se conectan a las rejas filtradoras con una tubería forzada (4). La velocidad del agua mueve las turbinas que a su vez accionan un generador produciendo energía eléctrica. Del generador se conduce la electricidad a los transformadores (9) donde se adecúa para la red de transporte (10). El agua que ya ha accionado las turbinas es devuelta al río por un canal de desagüe [11].

La inversión para una central con una potencia instalada de 1.500 MW es de aproximadamente 350 millones de euros. El rendimiento de las centrales hidroeléctricas en conversión de energía potencial a eléctrica, atendiendo a la ecuación (1) está sobre el 80%-85% [17].

2.3.3 Energía solar

El objeto de estudio de este trabajo es la mejora de la eficiencia de una célula fotovoltaica. Dicha célula fotovoltaica, que se encuentra integrada en una placa fotovoltaica, aprovecha la radiación solar, absorbiéndola y generando a partir de ella energía eléctrica. Las placas fotovoltaicas se agrupan en centrales fotovoltaicas para generar electricidad en grandes cantidades [11,27].

2.3.3.1 Centrales fotovoltaicas

Las centrales fotovoltaicas agrupan grandes números de placas fotovoltaicas para aprovechar la radiación solar incidente en el emplazamiento de la central en cuestión. La Fig. 14 presenta un esquema de una central fotovoltaica [11].

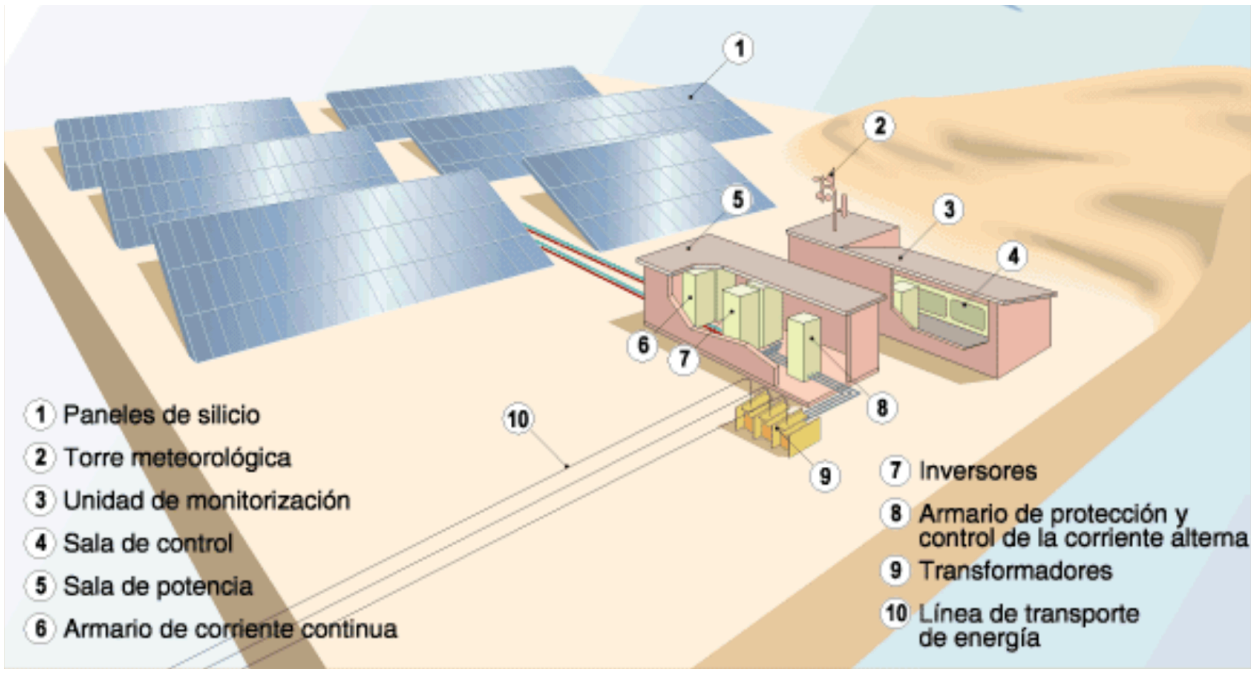

Figura 14. Componentes y funcionamiento de una central fotovoltaica.
Fuente: UNESA (http://www.unesa.es/sector-electrico/funcionamiento-de-las-centrales-electricas/1345-central-fotovoltaica)

El elemento básico de estas centrales es el conjunto de células fotovoltaicas, que son las que son capaces de captar la energía solar y convertirla a corriente eléctrica continua (DC) mediante el efecto fotoeléctrico. Las células fotovoltaicas están integradas en módulos que a su vez forman, en conjunto, los paneles fotovoltaicos de silicio (1) (Fig. 15). La generación de electricidad por parte de las células fotovoltaicas se ve afectada en gran medida por las condiciones climatológicas del entorno en que se hallen y son analizadas y predichas en la torre meteorológica [11,28].

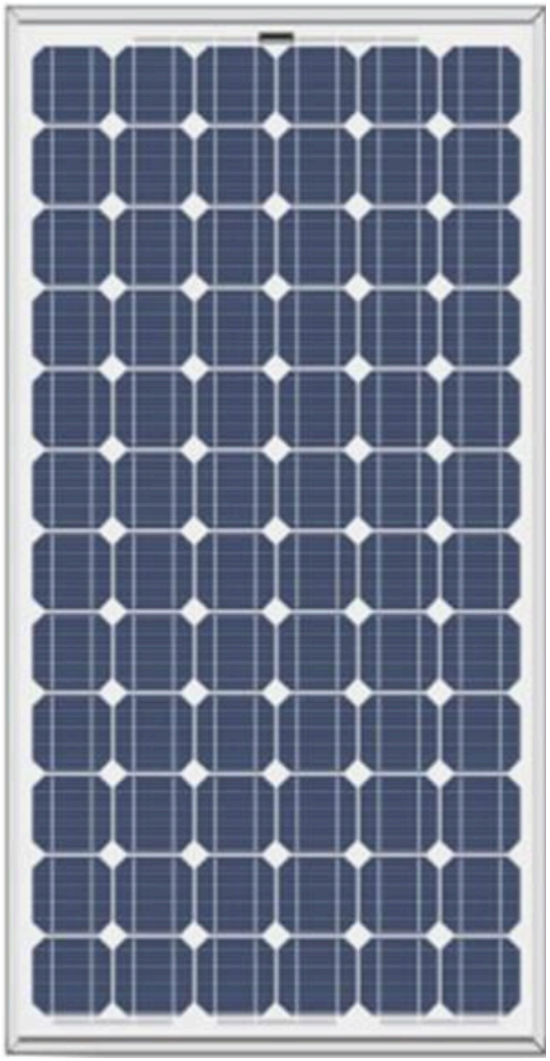

Figura 15. Placa solar de 180 W con 72 células (24 v).
Fuente (https://bateriasyamperios.com/shop/panel-solar-fotovoltaico-de-ocasion-165w-24v-copiar/)

La red de transporte de electricidad conduce la corriente en forma de corriente alterna (AC). Como se ha mencionado en el párrafo anterior, las células fotovoltaicas generan la electricidad en forma continua, por lo que es necesario transformarla en corriente alterna antes de ser conducida hacia la red de distribución. Dicha transformación se lleva a cabo a través de unos inversores (7) y se traslada consecuentemente a un armario de protección y control de corriente alterna (8). Finalmente, como ocurre en todas las centrales mencionadas en este trabajo, la corriente eléctrica es conducida hacia unos transformadores (9) que adaptan la tensión e intensidad de la misma a la de la red de distribución [29].

2.3.3.2 Células fotovoltaicas

Las células fotovoltaicas (PV) son las encargadas de la absorción de la radiación solar. Su uso general se explica a continuación [29,31].

Se utiliza una placa de sicilico puro para fabricar una célula PV, cuya parte superior se compone de una fina capa de dopantes “n”, negativo, típicamente de fósforo, y cuya parte inferior o base está formada por una pequeña cantidad de dopantes “p”, positivo, normalmente de boro. El lado de boro de la losa es 1.000 veces más grueso que el del fósforo. Tanto el fósforo como el boro son

materiales estructuralmente muy similares al material primario, silicio, teniendo el fósforo un electrón más, y el boro un electrón menos que este. Los dopantes ayudan a la creación de un campo eléctrico que fomenta la salida de los electrones cuando la luz solar choca con la célula PV. El silicio de tipo "n" no está cargado, pero tiene un exceso de electrones, por lo que algunos de ellos no están sujetos a átomos y se mueven libremente dentro de la capa. El silicio de tipo "p", con tendencia a atraer electrones, carácter positivo, pero no una carga positiva [29,30].

En el punto donde se encuentran los tipos "n" y "p", hay un flujo de electrones de la capa "n" a la "p" durante una fracción de segundo y forman seguidamente una barrera, denominada unión p-n, que impide el flujo de más electrones. Cuando los dos lados de la placa de silicio están dopados, existe una carga negativa en "p" y una carga positiva en "n" que se deben al movimiento de los electrones y a los "huecos" en la unión de ambos tipos. Este desequilibrio es el causante del campo eléctrico formado en la unión p-n [29,30].

Si la célula fotovoltaica está expuesta al sol, los fotones de luz colisionan con los electrones en la unión p-n, excitándolos y liberándolos de sus átomos. Estos electrones son atraídos por la carga positiva de la unión de silicio tipo n y repelidos por la carga negativa de la unión de silicio tipo p. La mayoría de las colisiones fotón – electrón ocurren realmente en la base de silicio [29,30].

Un cable conductor conecta el silicio tipo p a una carga eléctrica, y al silicio de tipo n, formando un circuito completo. A medida que los electrones libres son introducidos en el silicio de tipo n, estos se repelen por tener cargas similares, ya que el cable crea un camino para que los electrones puedan alejarse unos de los otros. Este flujo de electrones es una corriente eléctrica que viaja a través del circuito desde el silicio n hasta el silicio p. Además de los materiales semiconductores, las células solares disponen de una rejilla metálica superior u otro contacto eléctrico para recoger los electrones del semiconductor y transferirlos a la carga externa, y una capa posterior de contacto para completar el circuito eléctrico [29,30].

El proceso descrito puede comprenderse mejor con la ayuda de la Fig. 16.

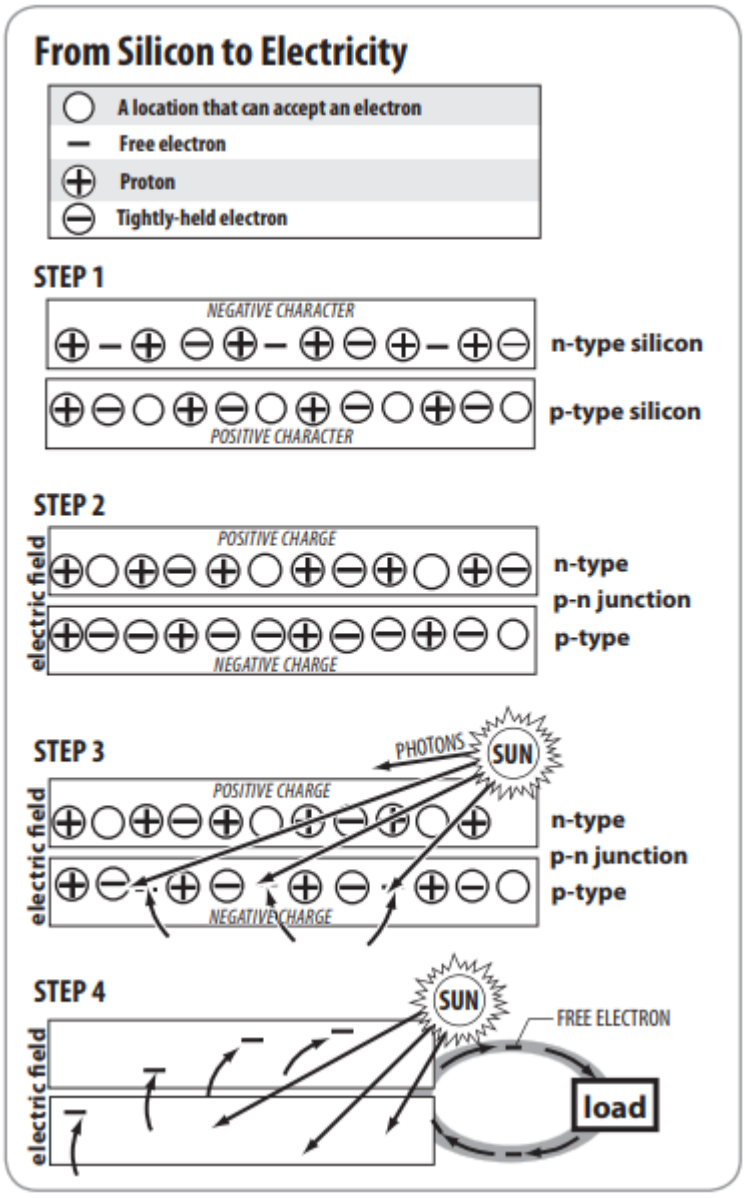

Figura 16. Funcionamiento de una célula fotovoltaica.
Fuente: (http://solardat.uoregon.edu/download/Lessons/Appendix_E_HowSolarCellsWork.pdf)

La eficiencia de una celda solar se puede dividir en eficiencia de reflectancia, eficiencia termodinámica, eficiencia de separación del transportador de carga y eficiencia conductiva. La eficiencia general es el producto de estas variables individuales [31].

La eficiencia de conversión de energía de una célula solar es un parámetro que se define por la facción de la potencia incidente convertida en electricidad. Una célula solar tiene una curva de eficiencia dependiente de voltaje, coeficientes de temperatura y ángulos de sombra permisibles [31].

Debido a la dificultad de medir estos parámetros directamente, se sustituyen otros parámetros: eficiencia termodinámica, eficiencia cuántica, eficiencia cuántica integrada, ratio V_{oc} y factor de relleno (fill factor, FF). Las pérdidas por recombinación y reflectancia constituyen una porción de la eficiencia cuántica, la relación V_{oc} y el FF [31].

El FF es la relación entre la potencia máxima real que se puede obtener y el producto de la tensión de circuito abierto y la corriente de cortocircuito, siendo un parámetro clave para evaluar el rendimiento. Las celdas con un alto FF tienen una baja resistencia en serie equivalente y una alta resistencia de derivación equivalente, por lo que una menor cantidad de corriente producida por la celda se disipa en pérdidas internas [31].

Los dispositivos de silicio cristalino de unión simple p-n se están acercando a la eficiencia de potencia límite teórica de 33.16%, [34,35] establecido como límite de Shockley-Queisser en 1961. En el extremo, con un número infinito de capas, el límite correspondiente es del 86% usando luz solar concentrada. [34]

En septiembre de 2015, Fraunhofer ISE anunció el logro de una eficiencia superior al 20% para las células epitaxiales. [34]

Para las células solares de película fina de triple unión, el récord mundial se establece en el 13.6%, en junio de 2015. [35]

En 2016, los investigadores de Fraunhofer ISE anunciaron una célula solar de triple unión GaInP / GaAs / Si con dos terminales que alcanzaban una eficiencia del 30.2% sin concentración. [35]

En 2017, un equipo de investigadores del Laboratorio Nacional de Energía Renovable (NREL), EPFL y CSEM (Suiza) reportaron eficiencias récord del 32.8% para dispositivos de células solares GaInP / GaAs de doble unión. Además, el dispositivo de doble unión se apiló mecánicamente con una célula solar de Si, para lograr una eficiencia récord de 35,9% para las células solares de triple unión. [35]

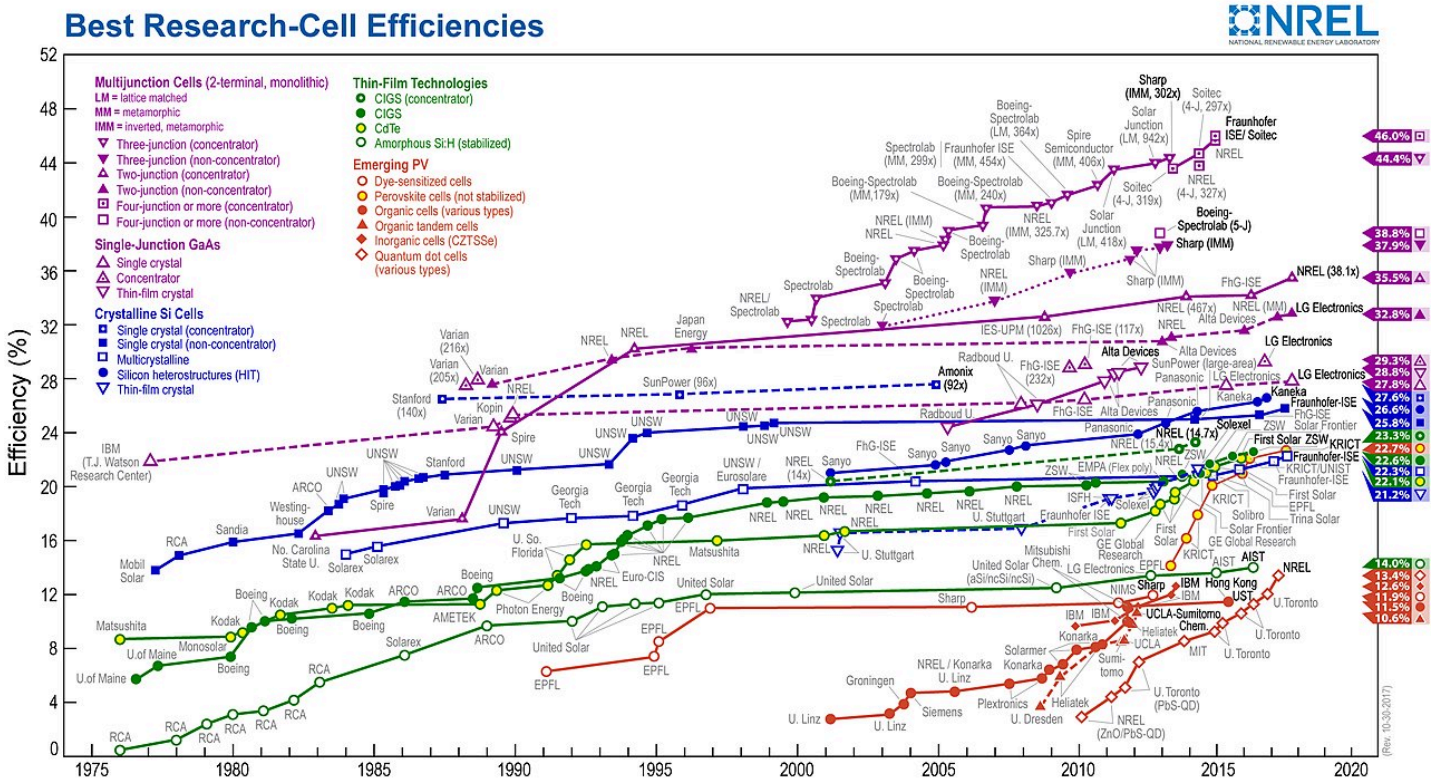

Figura 17. Eficiencia de conversión de energía de las células solares desde 1916.
Fuente: (National Renewable Energy Laboratory).

Las células solares se dividen tradicionalmente en tres generaciones.

- La primera generación se basa en placas de silicio y tienen un rendimiento del 15-20%. Son las células PV más extendidas del mercado y sus beneficios radian en su alto rendimiento y estabilidad. Su rigidez implica un alto uso de energía para su producción [36].

- La segunda generación tiene como objetivo reducir costos en materiales, por lo que reducen el uso del silicio en las células PV. Aunque estas células se pueden producir y comercializar, sin embargo, al estar relacionadas con procesos de vacios y altas temperaturas, la energía empleada para ello sigue siendo muy alta, y al utilizar materiales poco comunes, el precio es un factor limitante [36].

- Para las células solares de tercera generación se usan materiales orgánicos o polímeros. La tercera generación cubre costosas células multiunión de alto rendimiento que mantienen el record mundial de rendimiento en células solares. Un nuevo tipo de células solares son las Perovskita, con un rendimiento muy superior al 20% en áreas muy pequeñas. Una ventaja de las células solares poliméricas es que permite una producción sencilla, rápida y económica con materiales fácilmente disponibles y potencialmente económicos. Aunque el rendimiento y la estabilidad de las células solares de tercera generación son limitados con respecto a las primeras generaciones, tienen un gran potencial e interés en su investigación para los próximos años [36].

2.4 Energías renovables en España: Energía solar

En España, como se observa en el gráfico de la Fig. 18, las energías renovables participan activamente, en un alto porcentaje, en la generación de energía eléctrica. Las energías más importantes de la industria de las renovables en España son: eólica, hidroeléctrica y solar [32].

España es uno de lo líderes mundiales en instalación de centrales eólicas, ocupando el cuarto puesto en potencia instalada con 22.622 MW, datos de 2012, por detrás de China, Estados Unidos y Alemania [32].

El desarrollo de la energía eólica en el territorio español ha sido exponencial, como se muestra en la Fig.18 [32].

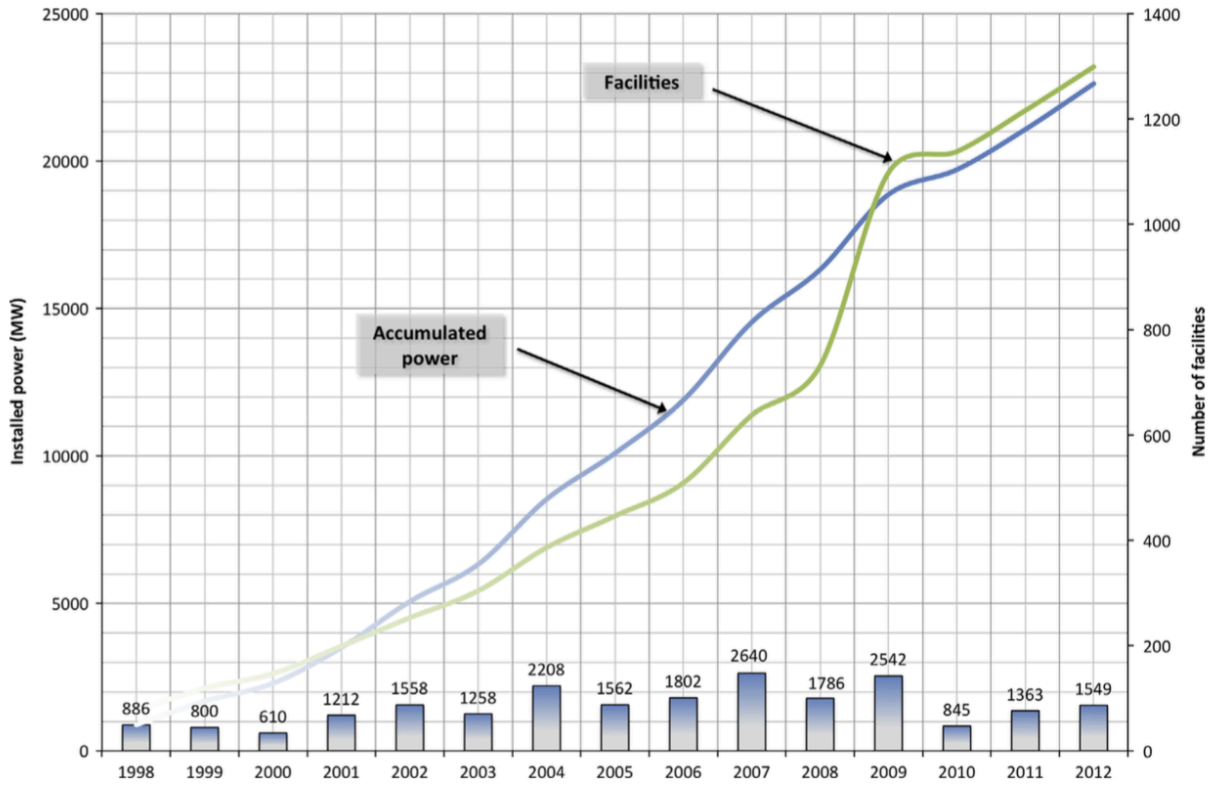

Figura 18. Desarrollo de la energía eólica instalada y número de instalaciones durante el período 1998-2012. Fuente: [32].

En la actualidad, hay en España unos 13.000 aerogeneradores de electricidad que suman una potencia instalada de 23.026 MW [32].

Para la energía hidroeléctrica influyen factores intrínsecos del país en cuestión. España tiene una base hidroeléctrica muy consolidada debido al terreno del país y al gran numero de embalses que hay en él. A nivel europeo, España ocupa el quinto puesto en el ranking de países con mayor potencia hidroeléctrica tras Suecia, Francia, Austria e Italia [32,36].

Lógicamente, la productividad de la industria hidroeléctrica depende del nivel de los embalses utilizados para la generación de energía eléctrica. En la Fig. 19 se representa la energía generada por año en España. Puede observarse cómo en 2010 por ejemplo, siendo un año muy lluvioso en España y estando por tanto los niveles de agua de los embalses muy altos, el aumento de la energía eléctrica generada con 6.748 GWh. Contrasta con los años 2011 y 2012, poco lluviosos en el país, con 5.280 y 4.620 GWh respectivamente [32].

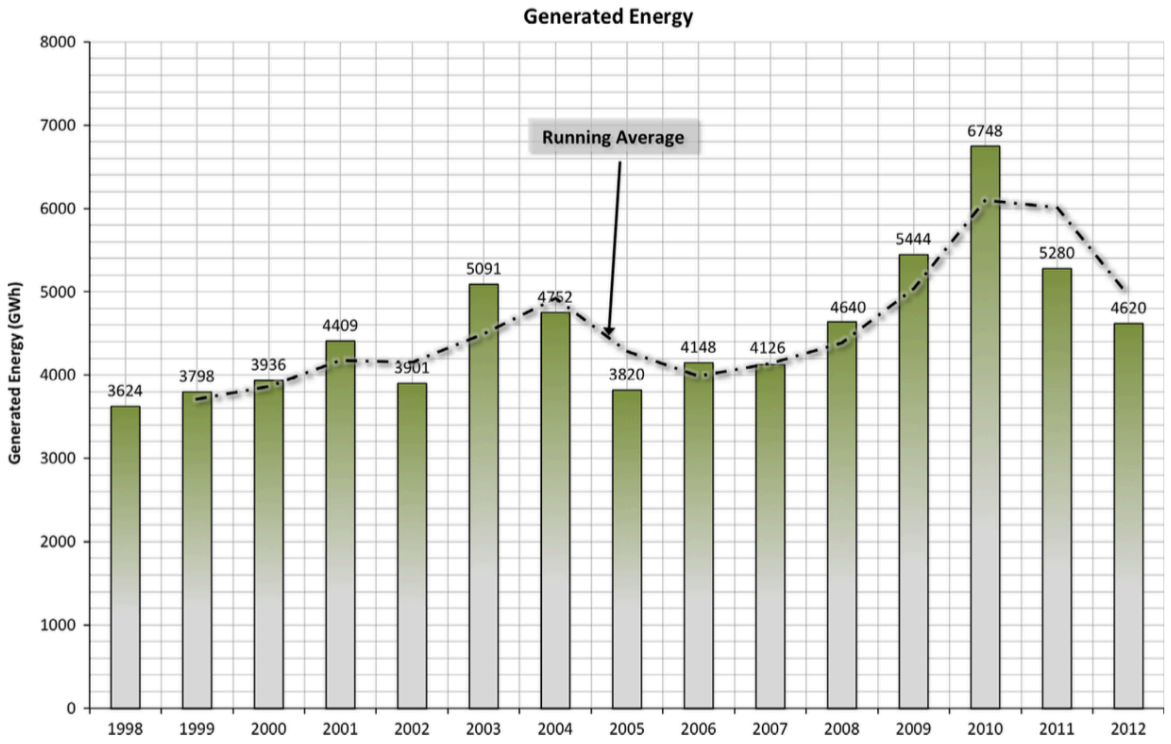

Figura 19. Desarrollo de la energía eólica generada durante el período 1998-2012. Fuente: [32].

En España hay unas 800 centrales hidroeléctricas con una potencia instalada de 17.792 MW.

En cuanto a la energía solar, las condiciones climatológicas de España son ideales para su aprovechamiento y fomento. Las horas de sol anuales varían desde las 1.600 horas en la zona norte hasta las 3.100 en el sur; la radiación, desde 1.5 kWh/m^2 en invierno a 7 kWh/m^2 en verano [32].

En el periodo 2008-2009, España se situó en el segundo puesto mundial en potencia instalada, sólo por detrás de Alemania, acumulando a año 2012 una potencia de 4.525 MW en 60.052 unidades fotovoltaicas (PV). A partir de 2011, entran en vigor en España una serie de restricciones que afectan al sector y lo hace entrar en crisis con perdidas del crecimiento de hasta el 30% en 2011, 2012 y 2013, y de un 10% en 2014. Los estudios decían que España debería estar produciendo 7.500 MW de potencia para el año 2020, pero la tendencia de cero crecimiento desde 2009 lo hacen prácticamente imposible [32].

Si se cumplen las nuevas expectativas para el sector español, se alcanzará una potencia instalada de 4.800 MW para finales de 2020 [32].

3 RAY TRACING

3.1 Antecedentes del Ray Tracing

Uno de los mayores objetivos de la computación gráfica a lo largo de muchos años ha sido el de encontrar un modo para crear imágenes "realistas". A la hora de mejorar la técnica, los investigadores en computación gráfica examinan el mundo a su alrededor y buscan la mejor imagen obtenida hasta la fecha, con la tecnología más actual. Para la mejora de imágenes computerizadas se introdujeron muchas características típicas de las escenas reales. Estos avances fueron posibles al observar que los objetos opacos ocultaban otros objetos, al ver que los objetos brillantes tienen más detalles o sabiendo que algunos objetos tienen un relieve superficial, permitiendo así desarrollar métodos de mejora de las imágenes computerizadas [39].

Uno de los primeros métodos de síntesis de imágenes comenzó con una idea de la física clásica en la que los físicos de la época trazaban en un papel el camino recorrido por la luz desde su origen, pasando por la lente y hasta más allá (Degli-Esposti et al., 2009). A este proceso de seguimiento de los rayos de luz se le llamó "Ray Tracing". La técnica de "Ray Tracing" se consideró un buen método para crear imágenes sintéticas, pero los ordenadores de la época (años 60) no permitían que las imágenes sintetizadas con "Ray Tracing" tuvieran un mejor aspecto que las creadas con otras técnicas más económicas [39].

A medida que los ordenadores se fueron volviendo más potentes, la técnica de "Ray Tracing" se fue desarrollando y volviéndose cada vez más atractiva hasta convertirse a día de hoy en una de las técnicas de síntesis de imágenes más valoradas y utilizadas [37,39].

3.2 Técnica del Ray Tracing

Como se ha comentado en el punto 3.1**,** la técnica de "Ray Tracing" consiste en realizar un seguimiento de la trayectoria de un rayo desde su origen hasta donde se estime oportuno. Para ello, es importante conocer los mecanismos de propagación de los rayos. Por simplicidad, se imagina un punto como origen de una multitud de rayos emanando de él. Se toma un único rayo y se basa la clasificación de cada tipo de rayo en su comportamiento:

- Rayos directos: si el rayo va directamente desde el punto de emanación hasta el punto de destino, se le denomina rayo directo y está relacionado con el mecanismo de propagación de la línea de visión.
- Rayos reflejados y transmitidos: si un rayo es reflejado (o transmitido) una o varias veces antes de alcanzar el punto de destino, se denomina rayo reflejado (o transmitido). Esto corresponde a la reflexión (o transmisión) de las ondas electromagnéticas entre diferentes medios. La dirección de propagación del rayo se determina mediante la ley de reflexión (ley de Fermat) y la magnitud del mismo se determina con las ecuaciones de Fresnel para distintas polarizaciones.
- Rayos difractados: los rayos difractados son mucho más complejos que los directos, reflejados y transmitidos. Un solo rayo incidente puede difractar una multitud de ellos, siendo el cálculo del coeficiente de difracción mucho más laborioso que el de los coeficientes de reflexión y transmisión. Existen diferentes métodos para el cálculo de los coeficientes de difracción que

podrían dar resultados diferentes. Las dificultades de la difracción de rayos se solucionaron parcialmente con el desarrollo de la GTD (Geometrical theory of diffraction) y la UTD (Uniform theory of diffraction). Una dificultad extra de la difracción aparece cuando hay múltiples aristas presentes, lo cual se ha resuelto recientemente para espacios 3D minimizando la distancia total de la trayectoria del rayo difractado cuando hay una secuencia de aristas.

- Dispersión: otro mecanismo de propagación es la dispersión por superficies rugosas, como la superficie del océano o fachadas de edificios [43, 44].

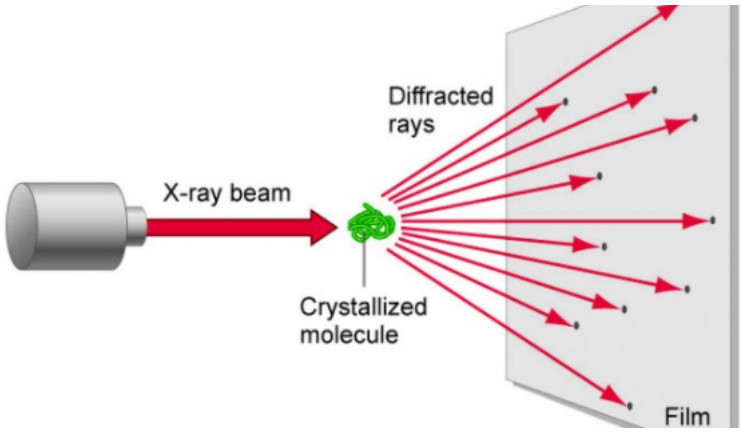

Figura 20. Difracción de rayos en Ray Tracing.
Fuente: (https://www.theskepticsguide.org/making-magnificent-movies-of-molecules-by-multitudes)

Un apartado clave en los métodos de Ray Tracing es el de determinar rayos desde su origen hasta un punto del espacio. En el espacio libre es sencillo, hay un solo rayo (rayo directo) que va en línea recta desde el origen hasta el punto de destino. Se van a examinar algunos de los algoritmos fundamentales en para el método de Ray Tracing:

1. Teorema general, principio de Fermat de reducción de tiempo: determina cómo un rayo encuentra su camino viajando de una localización a otra en un medio de propagación. Según este principio, el rayo tomará el camino que menos tiempo le haga emplear para llegar de un punto a otro. A partir de este método se pueden deducir las leyes para la reflexión, transmisión y difracción, a parte de el "método de la imagen", comúnmente utilizado para la determinación de la trayectoria de rayos reflejados [43, 44].

   - Método de la imagen: conocidos el punto (Tx) de origen y (Rx) de recepción, la trayectoria de un rayo reflejado en una superficie plana (Σ) se puede obtener fácilmente usando este método. Primero se obtiene el punto simétrico de Rx con respecto al plano Σ (Ri), se unen Tx y Ri en un segmento que contiene el punto Q de intersección con el plano Σ. De esta manera se halla la trayectoria del rayo definida por los puntos Tx, Q y Rx. El método de la imagen puede ser utilizado para determinar una trayectoria con múltiples reflexiones, pero a la hora de simularla computacionalmente puede resultar un método lento, por lo que el método "shooting and bouncing ray" (disparo y rebote de rayo) está más extendido en la práctica [43, 44].

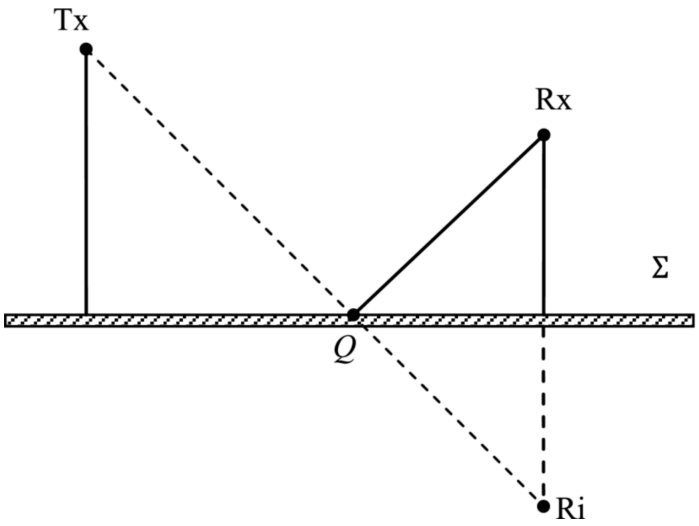

Figura 21. Método de la imagen. (Yun e Iskander, 2015)

2. Shooting and bouncing ray (SBR): la idea del SBR es la de trazar cada rayo lanzado desde su origen y determinar si alcanza el punto objetivo usando tres pasos: lanzamiento del rayo, trazado del rayo y recepción del rayo. Para el paso, lanzamiento del rayo, se suele requerir que todos los rayos que emanan del punto de origen estén uniformemente distribuidos de manera que porten una potencia similar para una fuente isotrópica. En la práctica este problema no causa realmente muchos problemas de precisión. Cuando un rayo es trazado, el segundo paso, puede ir directamente al punto de destino o puede ser reflejado y difractado varias veces antes de alcanzarlo. Si un rayo alcanza un objeto, se generará consecuentemente un rayo reflejado o difractado, lo que consume el 90% del tiempo de computación en los algoritmos de Ray Tracing no optimizados. El tercer paso es la recepción de un rayo. El rayo se asocia a un tubo de rayos con sección transversal creciente a medida que el rayo avanza. Cuando un tubo de rayos ilumina el punto de recepción, se recibe el rayo y es posible calcular la trayectoria seguida y comprobar que el punto de recepción está dentro del tubo de rayos [43].

- Método híbrido: a pesar de que el método SBR es fácil de implementar, en ocasiones no resulta tan preciso como el "método de la imagen", por lo que el método híbrido propone fusionar la rapidez computacional del SBR con la precisión de la trayectoria del rayo en el "método de la imagen". Primero se utiliza el SBR para determinar un rayo válido recibido en Rx y después se emplea el "método de la imagen" para calcular su trayectoria [43].

3.3 Ray Tracing en células fotovoltaicas

El empleo de la técnica de Ray Tracing en células PV se explica con la ayuda de un módulo como el de la Fig. 22. En la Fig. 22 a) se representa una célula PV en la que se distinguen el cristal, en la parte superior, una capa de acetato etil-vinilo (EVA) y el módulo de silicio; y, por otro lado, en la figura Fig. 22 b) en la que se distingue la separación entre células PV. [40].

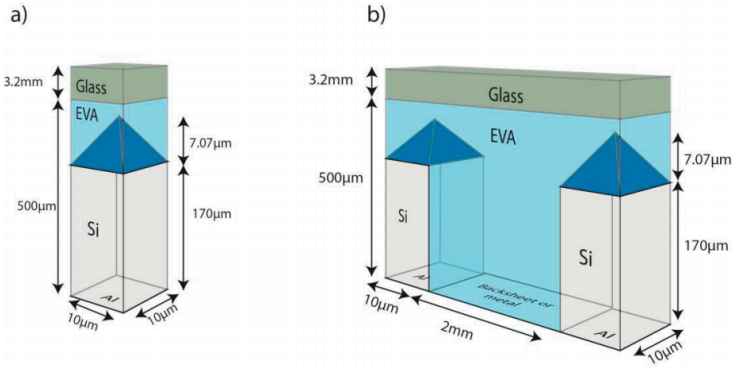

Figura 22. Dominios de simulación entre los que se desplazan los rayos en células fotovoltaicas.

(Holst et al., 2013)

La separación entre células es normalmente de unos 2 mm y contiene interconectores. La luz viaja unos centímetros mientras las células están cubiertas a través de unas pirámides en el rango micrométrico. Las simulaciones con Ray Tracing en módulos de células PV están determinadas por la estructura geométrica de las mismas en un amplio rango de dimensiones, que va desde micrómetros hasta centímetros. Por ello, el grado de complejidad de las simulaciones con Ray Tracing en módulos de células PV es muy superior al de células PV individuales [41,42].

Si la trayectoria de un rayo se iniciase en la parte superior de la célula PV, el rayo trazaría un recorrido conocido en el dominio de la célula PV, (Fig. 22 a)), donde probablemente rebotaría varias veces hacia delante y hacia atrás entre las paredes simétricas de la célula PV. Mientras tanto, se calcularía la posición global del rayo (r_g) y se observaría si el rayo permanece en la célula. Si r_g alcanza el borde externo de la pirámide de la célula PV, el trazado del rayo continúa automáticamente en un segundo dominio de simulación mostrado en la Fig. 22 b). En el caso de que r_g se introduzca de nuevo en el dominio de la célula PV a través de la zona piramidal, tras una serie de rebotes en la zona de separación entre células PV, el trazado podría continuar en el dominio de la Fig. 22 a) y así sucesivamente [40].

Actualmente, se ha demostrado que bajo el espectro AM1-5g, la eficiencia de un módulo c-Si (crystalline sillicon) se puede incrementar relativamente entre un rango de 0.5-1.5% mediante el uso de un encapsulante de silicona como sustituyente de EVA. Este aumento se debe principalmente a que los fotones de longitudes de onda inferiores a 400 nm son transmitidos por el silicio, pero bloqueados por EVA. El mayor aumento de eficiencia se observa para las células con una buena blue response (capacidad de absorción de longitudes de onda inferiores a 420 nm) y para las células de silicio con un índice de refracción más alto. Las placas de silicio muestran una ventaja óptica considerable con respecto a EVA cuando se calcula el espectro AM1-5g usando la técnica de Ray Tracing. Esto es debido a que los espectros de incidencia suelen ser más "azules" (menor longitud de onda) que el espectro AM1-5g, especialmente en verano y en días nublados. En estos casos, una mayor proporción de fotones tiene una longitud de onda inferior a 400 nm [38, 41].

Estos estudios se han limitado al espectro AM1-5g, altamente influyente en el campo de estudio de las células PV y sus módulos. Aunque un espectro estandarizado es primordial para la comparación de células y módulos, el espectro AM1-5g casi nunca representa el espectro en el campo de estudio. En la práctica, el espectro incidente varía de un sitio a otro a lo largo de cada día y año debido a los cambios en la masa del aire, la turbidez, el agua precipitable, las nubes y el

albedo. Los cambios anuales en el espectro causan grandes variaciones en la eficiencia de los módulos de capa delgada, especialmente aquellos con células en tándem [38, 41].

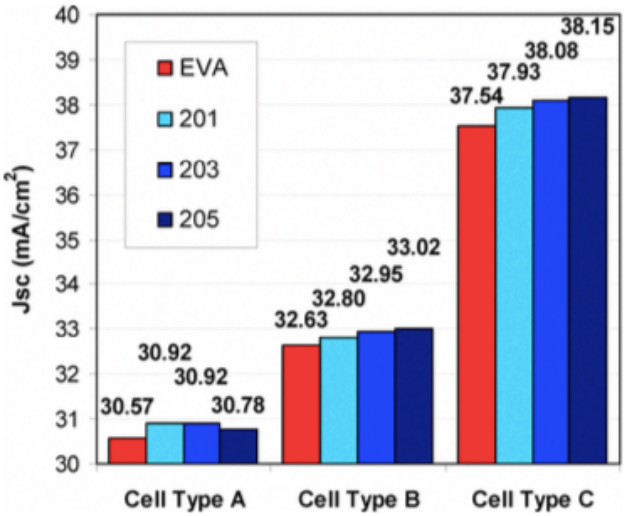

Figura 23. Diagrama transversal de un módulo fotovoltaico convencional (a escala) y los mecanismos de pérdida óptica correspondientes**.** (McIntosh et al., 2009)

Una evaluación óptica precisa de un módulo PV no es trivial. Como se ilustra en la Fig. 24, los rayos incidentes se reflejan desde las interfaces de vidrio de aire (1), del encapsulante de vidrio (3), de la célula encapsulante (6) y de la hoja posterior encapsulante (8); y en los dos últimos casos, el reflejo es a menudo difuso, lo que hace que parte de la luz reflejada se refleje internamente en la interfaz vidrio-aire y luego se devuelva a las celdas. Además, los rayos incidentes se absorben en el vidrio (2), en el encapsulante (4), en el revestimiento antirreflejante de la célula o en los dedos metálicos (5) y en la lámina posterior (7). Estas ocho interacciones dependen de la longitud de onda incidente y del ángulo de incidencia de la luz. Por lo tanto, una evaluación óptica completa es apoyada por la aplicación del Ray Tracing [38, 40, 41].

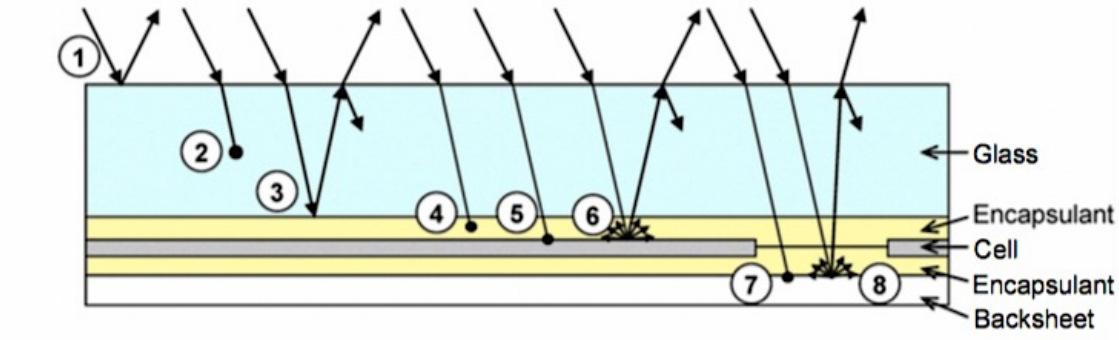

Figura 24. Resultados del estudio de Ray Tracing trazados según la densidad de corriente de cortocircuito (J_{sc}) del módulo en función de cada tipo de célula y encapsulante. (McIntosh et al., 2010)

McIntosh et al. (2009) utilizaron la simulación de Ray Tracing para cuantificar las pérdidas ópticas de tres módulos fotovoltaicos convencionales ((A) planar multi-cSi screen-printed cells, (B) textured mono-cSi screen-printed cells, and (C) textured mono-cSi rear-contact cells). Observaron que, para un módulo encapsulante de silicio en lugar de EVA, hay un aumento de 0.9-1.6% en la Jsc (short-circuit current density) del módulo (Fig. 24). Este aumento es principalmente debido a

la transmisión de luz de longitud de onda corta («420 nm), y es por tanto mayor cuando se utiliza con vidrio de baja absorción y células con un IOE (intensity of emission) alto a longitud de onda corta, como las células de contacto posterior (tipo C). La ventaja sigue siendo significativa, sin embargo, incluso cuando se trata de las células más modernas screen-printed multi-cSi (tipo A) [40, 41].

Además, McIntosh et al. (2009) también cuantificaron la absorción en vidrio Starphire® de bajo contenido en hierro (1.5%), en EVA (0.2%) y en silicio (0.1%) a longitudes de onda mayores de 420 nm. Esta absorción en el vidrio depende en gran medida de la concentración de hierro y podría ser significativamente mayor para otros tipos de vidrio, incluso aquellos clasificados como bajos en hierro. Finalmente, se ha descrito la importancia del índice de refracción del encapsulante. En el intervalo examinado, tiene un pequeño efecto sobre la reflexión en la interfaz del encapsulante de vidrio y un pequeño efecto en la interfaz de la célula encapsulante. Por lo tanto, el silicio es más preferible a EVA cuando las células tienen una IOE alta en longitud de onda corta, cuando la silicona tiene un alto índice de refracción, y cuando hay una gran área expuesta de la lámina de respaldo [40, 41].

4 CÓDIGO Y SIMULACIONES

4.1 Explicación del código, muestra de variables y parámetros más importantes

La elaboración del código usado para las simulaciones de Ray Tracing está basado en el código desarrollado en Visual Basic por la empresa australiana PV Lighthouse.

El código está escrito en MATLAB y consta de 1.494 líneas en su parte principal y se complementa con otras 18 funciones empleadas en el código principal para realizar cálculos intermedios.

La utilidad de este programa consiste en la posibilidad de realizar simulaciones de Ray Tracing permitiendo la modificación de numerosos parámetros y el estudio de la mejora de la eficiencia en la absorción de energía solar según la elección de estos parámetros. Esta característica hace que sea un programa de simulación versátil y útil para el propósito del proyecto. Además, el programa incluye varias muestras de espectros lumínicos en distintas zonas geográficas y estaciones del año que permiten elegir unas configuraciones u otras según la conveniencia en cada caso. Las muestras de los espectros están recogidas en unos archivos de texto .txt que se adjuntan como material de este proyecto junto al código principal y las funciones en MATLAB.

4.1.1 Datos y parámetros de entrada disponibles

Los espectros disponibles son:

- AM0
- AM1-5d
- AM1-5g
- 8 de febrero de 1961 a las 12 pm en Alamosa, Estados Unidos
- 4 de mayo de 1967 a las 12 pm en Sacramento, Estados Unidos
- 24 de junio de 1976 a las 12 pm en Phoneix, Estados Unidos
- 4 de julio de 1983 a las 12 pm en Brownsville, Estados Unidos
- 6 de diciembre de 1985 a las 12 pm en Buffalo, Estados Unidos

AM0 y AM1-5 son los espectros estandarizados según el coeficiente de masa de aire. El AM0 correspondería al espectro con 0 atmósferas; el AM1, con una atmósfera. El espectro AM1-5 (AM1.5) correspondería al espesor de 1.5 atmósferas. Como la luz solar no incide todo el año con la misma inclinación, el espectro AM1.5 es una buena aproximación de la media anual en latitudes templadas como Europa, China, Japón, Estados Unidos y algunas latitudes del hemisferio sur.

Aparte del espectro, el programa permite la simulación de Ray Tracing en una célula fotovoltaica compuesta por hasta 6 capas (contando la capa externa (6) y la capa de la propia célula (1)). Para estas capas se puede elegir el grosor y el material del que están hechas. Los datos de los materiales, al igual que los de los distintos espectros están recogidos en ficheros de textos .txt y puede seleccionarse cualquiera de la siguiente lista:

- Aire
- EVA
- Vidrio
- Polimetilmetacrilato (PMMA)
- Silicio
- PECVD
- Óxido de silicio (SiO_2)
- Óxido de titanio (TiO_2)
- Óxido de cinc (ZnO)

El grosor de estas capas se utiliza en centímetros en el programa. También es posible elegir 5 películas que recubrirían, si se desea, cada una de las capas (excluyendo la externa). Para estas películas también es posible elegir el grosor, en este caso en nanómetros, y el material del que están hechas, que son los mismos seleccionables para las capas.

En cuanto a las características físicas de las células, es posible elegir varios parámetros. Pueden elegirse las dimensiones de la célula (longitud, anchura y radio) y la separación entre células en coordenadas x, y. Con respecto a la superficie de la misma pueden elegirse el número de "busbars" y su anchura, y, la longitud de "fingers" y su anchura.

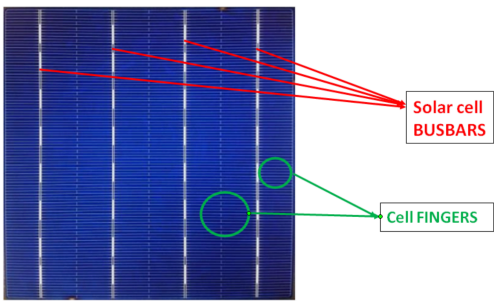

Figura 25. "Busbars" y "fingers" de una célula fotovoltaica.

También es posible elegir dos parámetros eléctricos de la célula fotovoltaica; la tensión a circuito abierto (Voc) y el factor de llenado (FF, fill factor). Ambas se relacionan entre sí, ya que, el factor de llenado es la relación entre la potencia máxima partida del producto de Isc (intensidad de cortocircuito) y Voc.

Aparte se pueden configurar algunas entradas adicionales.

- La localización inicial del rayo puede ser aleatoria, o bien introducirse manualmente en coordenadas x, y.
- El tipo de iluminación, que puede ser directa, ángulo cenital 180º, ángulo acimutal 270º; o difusa, ángulos aleatorios con el ángulo cenital siempre entre 90º y 180º.
- Puede elegirse incluir o no las pérdidas del módulo en sus extremos.
- Los datos de la reflexión de la célula ARC puede calcularse o tomarse de un fichero .txt

previamente cargado.

- Las longitudes de onda final e inicial
- El número de rayos en la simulación, el número máximo de rebotes de cada rayo y la intensidad mínima (Imin) a partir de la cual se considera despreciable la energía restante del rayo.

Una vez seleccionados todos los parámetros editables del código se prodece a la ejecución del mismo.

4.1.2 Funciones del programa

El programa está compuesto de un total de 18 funciones que complementan al código principal. Una gran parte de estas funciones, 8 en concreto, realizan simplemente operaciones matemáticas para las cuales MATLAB no tiene una función definida, por lo que sólo serán mencionadas:

- ComplexArcSine
- ComplexArcCosine
- ComplexDivision
- ComplexLogarithm
- ComplexMatrixMultiplication
- ComplexMultiplication
- ComplexSine
- ComplexSquareRoot

El resto de las funciones que se adjuntan tienen un papel más importante en el cálculo de variables intermedias dentro del código principal, son estas 10:

- **Calculate_rt_Amplitude_Coefficients**: Esta función no se usa directamente en el código principal, si no que forma parte de una función que se explicará a continuación llamada **NewCalculateRAT**. La función **Calculate_rt_Amplitude_Coefficients** recibe como entradas dos ángulos, el de incidencia y el de transmisión; y dos índices de refracción que variarán según el material con el que estemos trabajando y que habrán sido cargados previamente por la función **LoadRIdata.** Las salidas serán los coeficientes de reflexión y transmisión de TE (polarización transversal eléctrica) y TM (polarización transversal magnética).
- **Calculate_thinfilm_rAt_coefficients**: Esta función tampoco forma parte del código principal, si no que también se encuentra dentro de **NewCalculateRAT**. En éste, la función también calculará los coeficientes de reflexión y transmisión de TE y TM, pero en este caso también se calculan los de absorción. Sólo se utiliza cuando hay películas presentes sobre la superficie de las capas. Como parámetros de entrada recibe el número de películas, se hace de una en una por lo que será siempre 1; el espesor de dicha película, los datos de índices de refracción del material del que esté hecha la película, el ángulo de incidencia thetasup y la longitud de onda con la que se esté trabajando en ese instante.
- **CalculateNewSuv**: Suv es el vector unitario perpendicular al plano de incidencia. Después de cada rebote del rayo al que se le esté aplicando la técnica de Ray Tracing, esta función se encarga de calcular el nuevo vector Suv que precede al del rebote anterior. La función recibe como entradas el número de rebotes que lleva el rayo, las componentes x, y, z del vector

unitario del rayo estudiado, la variable **dotproduct**, que advierte si el rayo ha alcanzado al plano; el vector unitario del plano y el número de plano al que intersecta según una estructuración que se verá más adelante. Las salidas de la función serán el nuevo **Suv** y las nuevas componentes x, y, z del vector del rayo.

- **DetermineWavelengthBin**: Bin son los compartimentos de la célula solar donde se almacenará la energía captada de los distintos rayos solares que lleguen al módulo. La función **DetermineWavelengthBin** determina para cada longitud de onda de cada uno de los rayos de la simulación, en el caso en que la longitud de onda no sea constante, en qué compartimento o **Bin** se almacenará la energía absorbida. La función recibe como entradas la longitud de onda del rayo en cuestión, las longitudes de onda del primer y último Bin de la célula, el intervalo de longitudes de onda entre cada Bin y el número total de Bins. Como salida, la función devuelve el Bin correspondiente a la longitud de onda en la que se esté simulando.

- **InterpolateData**: Esta función se encarga de interpolar un array o conjunto de datos para devolver uno adecuado según las dimensiones del array y los valores de sus componentes. La función está diseñada para la interpolación de arrays de 2, 3 y 4 dimensiones, si bien en este código sólo se usan arrays de 2 y de 3 dimensiones. Como entradas, **InterpolateData** recibe en primer lugar el array de datos, el número de dimensiones de dicho array, una variable **x** que otorga el acceso al primer índice de cada dimensión (una vez determinada ésta) a otras dos variables internas de la función, dos variables que se usan para acceder a los elementos de los arrays de 3 y 4 dimensiones; y dos constantes conocidas que darán acceso al segundo índice de cada dimensión. Para mayor comprensión de la cabecera de la función se adjunta la **Figura 26.** Como salida se obtiene el valor interpolado del array de datos original.

```
%%                FUNCIÓN INTERPOLATEDATA (1895-2000)                %%

function [y]=InterpolateData(a, dims, x, dim1, dim2, xCol, yCol)

% a: conjunto de datos? ;  dims: dimensiones en el conjunto de datos ;  x: valor de x ;  y: valor de y (devuelto) ;
% dims1 & dims2 son las variables para las dimensiones 1 & 2 (si fuese necesario)

% Asumimos que:
% 1) x aumenta monótonamente
% 2) no hay elementos en el conjunto de datos en los que x esté vacía excepto para una secuencia continua de elementos vacíos ¿en la parte alta del conjunto de datos?

LL=1; % ignora entradas en el elemento 0. (No se si afectará al resultado en MATLAB)

switch dims
    case 2
        UL=length(a);    % UBound(a,1) en excel. Límite superior es igual al tamaño del conjunto
    case 3
        UL=length(a{dim1});   % UBound(a,2) en excel. Límite superior es igual al tamaño del conjunto
        %     case 4      % No hay ningún caso en todo el código, si da problemas lo suprimiré.
%         UL=max(a{3});       % UBound(a,3) en excel. Límite superior es igual al tamaño del conjunto
end

while (UL-LL)~=1
    switch dims
        case 2
            xLL=a(LL,xCol);
            xUL=a(UL,xCol);
        case 3
            xLL=a{dim1}(LL,xCol);
            xUL=a{dim1}(UL,xCol);
%         case 4                                    No hace falta, no se usa para nada
%           xLL=a(dim1,dim2,LL,xCol);
%           xUL=a(dim1,dim2,UL,xCol);
    end

    if x<=xLL
        switch dims
            case 2
                y=a(LL,yCol);
            case 3
```

InterpolateData Ln 114 Col 24

Figura 26. Cabecera y principio de la función **InterpolateData**.

- **LinearInterpolation**: Realiza una interpolación dentro de la función **InterpolateData** en el caso singular en que dicha interpolación fuese lineal. Recibe como entradas 4 elementos del array en cuestión además de la variable **x** recibida originalmente por **InterpolateData** y devuelve el valor interpolado linealmente.

- **LoadRIdata**: Recibe como entradas el número de capa de la que se quiere recopilar la información y el nombre del material del que está hecha la capa. La función compara el nombre de material con todos los que se tienen en la base de datos de los demás materiales en archivos de texto .txt. Una vez determinado el archivo correcto se introducen los datos de índices de refracción del material en dos arrays de tipo *Cell* de MATLAB. Este tipo de arrays permite almacenar vectores y matrices de distintos tamaños, lo que resulta muy útil en este caso. La

función devuelve como salidas dos arrays de tipo *Cell* en los que se almacenan todos los datos que contenía el archivo .txt.

- **LoadSpectrum**: Sólo recibe una entrada, el nombre del espectro luminoso que se va a utilizar para el estudio de Ray Tracing. Análogamente con **LoadRIdata** se extrae la información necesaria del archivo .txt correspondiente al nombre de espectro recibido. A partir de aquí, la función realiza una serie de cálculos para devolver cuatro elementos de salida: las primera y última longitudes de onda del espectro, la intensidad espectral en el fichero y el array de los espectros normalizados con la intensidad espectral del fichero.
- **NewCalculateRAT**: Esta función se encarga de calcular las nuevas transmisión, reflexión y absorción después de cada rebote. La función utiliza, como se explica anteriormente, dos funciones en su interior para calcular variables intermedias que se usan para el propósito final de **NewCalculateRAT**. Como entradas, la función recibe el número de rebotes hasta el momento del rayo al que se le hace el estudio de Ray Tracing, el grosor de película de estudio (si la hubiere), los datos de índices de refracción del material utilizado, el ángulo incidente del rayo, la longitud de onda del mismo, la intensidad restante después de los rebotes y el vector de polarización **E** en coordenadas s, p. Como salidas se obtienen la Erefl, Etrans y la reflexión, transmisión y absorción de este rebote.
- **SetPlaneUnitVector**: Esta función no tiene entradas, por lo que en realidad es prescindible y podría escribirse en el código principal, no obstante, se ha hecho así. La función devuelve los vectores unitarios de los planos que componen el módulo de estudio y que se clasifican así: Plano 1, fondo (célula); Plano 2, arriba (exterior); Plano 3, Sur; Plano 4, Norte; Plano 5, Este; Plano 6, Oeste.

4.1.3 Funcionamiento general del programa

El objetivo final del programa es comprobar como va variando la eficiencia del módulo según se cambien los parámetros editables que se mencionaron anteriormente. Para ello, se realizan 50.000 simulaciones de Ray Tracing distintas, cada una con sus propias características espectrales según los datos obtenidos en los ficheros .txt. Para cada rayo se calcula su Bin correspondiente, se calcula su longitud de onda su ángulo de incidencia y los vectores asociados al primer rebote nuv, suv y kuv (kuv es el vector unitario del rayo).

Una vez calculado esto se simulan los rebotes del rayo dentro del módulo con dos líneas rojas; la intensidad del rayo no puede bajar de cierta cantidad Imin o se parará el proceso de Ray Tracing de ese rayo y se saltará al siguiente, y el número de rebotes de un solo rayo no puede superar el número preestablecido **bmax**.

Dentro de cada rebote se calcula la perdida de energía y su destino y se almacena en las variables correspondientes. Se calculan los nuevos vectores nuv, suv y kuv y se comprueban las dos condiciones de continuidad.

Acabados los rebotes del rayo se almacenan los datos de energía dentro de su variable y Bin correspondiente.

Por último, se calcula la cantidad total de energía absorbida, transmitida o reflejada por cada uno de los elementos que componen el módulo de estudio y se calcula a partir de los parámetros eléctricos la eficiencia general de este. Se calcula también al final la EQE (Eficiencia cuántica externa) del módulo y se compara con la IQE (Eficiencia cuántica interna). La eficiencia cuántica

corresponde al número de electrones producidos en el circuito externo a la celda solar por cada fotón del espectro incidente.

4.1.3.1 Cálculos matemáticos para un solo rayo

En este apartado se explican los fundamentos matemáticos utilizados para el cálculo de Ray Tracing para un solo rayo. Para ello se explica algo más de detalle este trozo de código. Esta es la parte más importante del código y comprende desde la línea 509 hasta la línea 1437.

Lo primero que se hace es determinar la longitud de onda del rayo en cada caso. Si el nombre del fichero de espectro es "Single wavelength", la longitud de onda será siempre la misma para todos los rayos; si no, utilizando la función **InterpolateData**, se interpolará el fichero de datos del espectro en cuestión y se determinará la longitud de onda correspondiente que variará gracias a un valor aleatorio que recibe por referencia la función. Seguidamente se interpolan también los datos de los materiales de cada una de las capas y películas (si existen) para determinar los parámetros ópticos *n* y *k* de cada una de ellas. Estos parámetros constituyen el índice de refracción y el coeficiente de extinción de cada material respectivamente. En nuestro caso concreto, obtenemos estos parámetros interpolando de la base de datos del archivo .txt del material correspondiente, sin embargo, el cálculo teórico de estos dos parámetros se obtiene mediante las relaciones Kramers-Kronig que estabelecen una relación entre ambos parámetros. Las ecuaciones de las relaciones discretizadas son:

$$n(\omega_i) = 1 + \frac{2}{\pi} \sum_{j \neq i} \frac{\omega_j}{\omega_j^2 - \omega_i^2} \cdot k(\omega_j) \cdot \Delta\omega_j \quad (2)$$

$$k(\omega_i) = -\frac{2 \cdot \omega_i}{\pi} \sum_{j \neq i} \frac{\omega_j}{\omega_j^2 - \omega_i^2} \cdot n(\omega_j) \cdot \Delta\omega_j \quad (3)$$

$$\omega_i = \frac{C_0}{2 \cdot \pi \cdot \lambda_i} \quad (4)$$

Una vez calculados los parámetros ópticos, se determina el ángulo de incidencia del rayo en el módulo, que puede ser aleatorio o constante según se haya configurado anteriormente. A partir de este punto se determina la posición inicial del rayo y comienza el seguimiento por Ray Tracing del mismo empezando por el primer rebote.

En el primer rebote se establece el vector **s** perpendicular al plano de incidencia, y se inicializan los vectores de polarización **Es** y **Ep** con las componentes reales aleatorias y las imaginarias igual a cero. Se cargan los parámetros ópticos que se interpolaron anteriormente en caso de existir capa o película y se llama a la función **NewCalculateRAT**. Esta función, como se explica anteriormente, se encarga de calcular los nuevos valores de los vectores de polarización y de la transmisión, absorción y reflexión después del rebote con el plano de incidencia.

Dentro de **NewCalculateRAT** hay dos maneras de calcular los parámetros de salida de la función dependiendo si hay o no película presente:

- Con película presente: Se utiliza la función **Calculate_thinflim_rAt_coefficients** para calcular los coeficientes de reflexión, transmisión y absorción de TE y TM. Los fundamentos matemáticos de esta función encuentran su base en las ecuaciones de Fresnel.

- Sin película presente: Se utiliza la función **Calculate_rt_Amplitude_Coefficients** en este caso. Aquí no se calcularía el coeficiente de absorción sino los de reflexión y transmisión exclusivamente. De nuevo, los fundamentos matemáticos de la función están basados en las ecuaciones de Fresnel.

$$rTE = \frac{\cos(\theta) - \sqrt{n^2 - sen^2(\theta)}}{\cos(\theta) + \sqrt{n^2 - sen^2(\theta)}} \quad (4)$$

$$rTM = \frac{-n^2 \cdot \cos(\theta) + \sqrt{n^2 - sen^2(\theta)}}{n^2 \cdot \cos(\theta) + \sqrt{n^2 - sen^2(\theta)}} \quad (5)$$

$$tTE = \frac{2 \cdot \cos(\theta)}{\cos(\theta) + \sqrt{n^2 - sen^2(\theta)}} \quad (6)$$

$$tTM = \frac{2 \cdot n \cdot \cos(\theta)}{n^2 \cdot \cos(\theta) + \sqrt{n^2 - sen^2(\theta)}} \quad (7)$$

$$n = \frac{n_{transmitido}}{n_{incidente}} = \frac{n_2}{n_1} \quad (8)$$

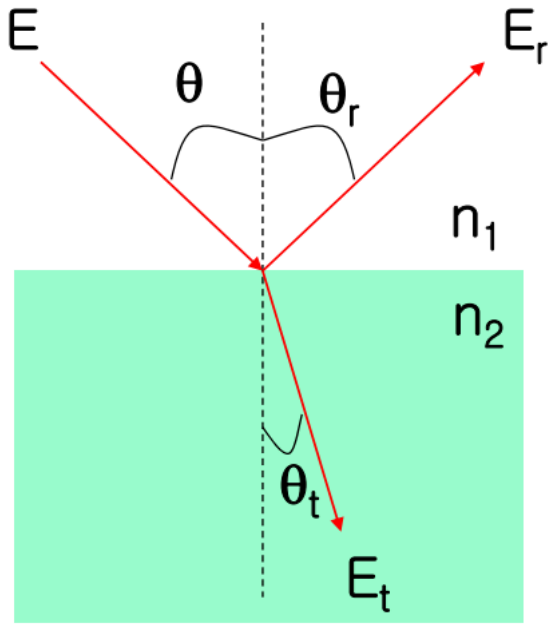

Figura 27. Ilustración de apoyo para la comprensión de las ecuaciones de Fresnel.

Una vez hecho esto, ya se tienen los vectores de polarización **Es** y **Ep** además de la transmitancia, absorción y reflexión del primer rebote. Sólo queda actualizar la intensidad que aún le queda al rayo transmitido. Esto lo averiguamos con una simple operación, el producto de la intensidad previa (en este caso la inicial) y la transmisión recién calculada con **NewCalculate_RAT**.

$$I_b = I_{b-1} \cdot trans \quad (9)$$

Hecho esto queda calcular el ángulo de salida (ángulo de transmisión), actualizar el ángulo **theta** de incidencia mediante este ángulo de transmisión y actualizar los vectores **kuv** (unitario del rayo) y **suv**.

A partir de aquí se entra en un bucle en el que se analizan los sucesivos rebotes del rayo.

Se resetean las componentes normales al nuevo plano de incidencia. Se determina el **Bin** correspondiente mediante la función **DetermineWaveLengthBin** y se asegura que el vector unitario del rayo no sea paralelo a los planos **x, y, z** para evitar errores.

El siguiente paso es la determinación de la posición del rayo en coordenadas x, y, z. Para ello hay que determinar en primer lugar con cuál de los 6 planos del módulo intersecta el rayo. Esto se comprueba determinando si la suma del producto de las componentes de **kuv** con las **nuv** de cada plano da un resultado negativo o positivo. Si da negativo, es que ha intersectado con el plano en cuestión y se determina la posición exacta del rayo. Una vez se tiene la posición puede calcularse la distancia recorrida por el rayo de un punto a otro en este rebote.

Si el plano en el que rebota el rayo es el plano del fondo, hay que determinar si la intersección es interna, externa o en el metal de detrás. En este plano se encuentra la célula fotovoltaica por lo que es vital para el programa saber qué pasa en la intersección de ese plano.

El programa realiza también un recuento de las pérdidas por absorción, transmisión y reflexión en el resto de las superficies para dar unos resultados más completos. Es lo que se hace en los siguientes apartados. Aparte, también se actualizan la transmisión y la intensidad que queda después de cada rebote, de manera análoga al rebote 1.

Entonces, en función de el plano de incidencia, de la región de incidencia en caso de ser plano 1 (se considerarían ganancias, **y lo más interesante de todo por ser la energía aprovechable**, la región 0, rebote interno en el plano bajo), y del tipo de radiación emitida se van calculando los sucesivos vectores de polarización **Es** y **Ep**, los vectores **kuv** y **suv** y se almacenarían los datos de interés, como pérdidas y ganancias por transmisión, reflexión y absorción de cada una las capas y planos que componen el módulo.

Como se dijo anteriormente, el número de rebotes tiene dos líneas rojas. La primera y principal línea roja es la energía mínima del rayo. La energía mínima del rayo es el límite que elegimos y a partir del cuál consideramos despreciable a efectos de contabilización de energía que pueda aportar dicho rayo después de un número **b** de rebotes. Una vez llegados a este valor de energía, se deja de tener en cuenta este rayo por lo que se saldría del bucle de contabilización de rebotes y se simularía el siguiente rayo.

La segunda línea roja es el número máximo de rebotes que establecemos nosotros mismos para que, a efectos de programación, la simulación no resulte tan pesada. Se comenta que es más importante la primera línea roja porque es la que normalmente limitará la contabilización de la energía aprovechable de un rayo y por tanto la que más influirá en el resultado final. En este trabajo se ha considerado que por debajo de $\mathbf{10^{-3}\ W/cm^2}$**.**
Una vez se ha llegado al máximo número de rebotes o al mínimo valor de intensidad del rayo, se simula el siguiente rayo, y así sucesivamente hasta llegar al número máximo de rayos seleccionados (en nuestro caso **50.000**).

En el siguiente diagrama de flujos se aprecia la estructura general del bucle de simulación de rayos descrito en este punto:

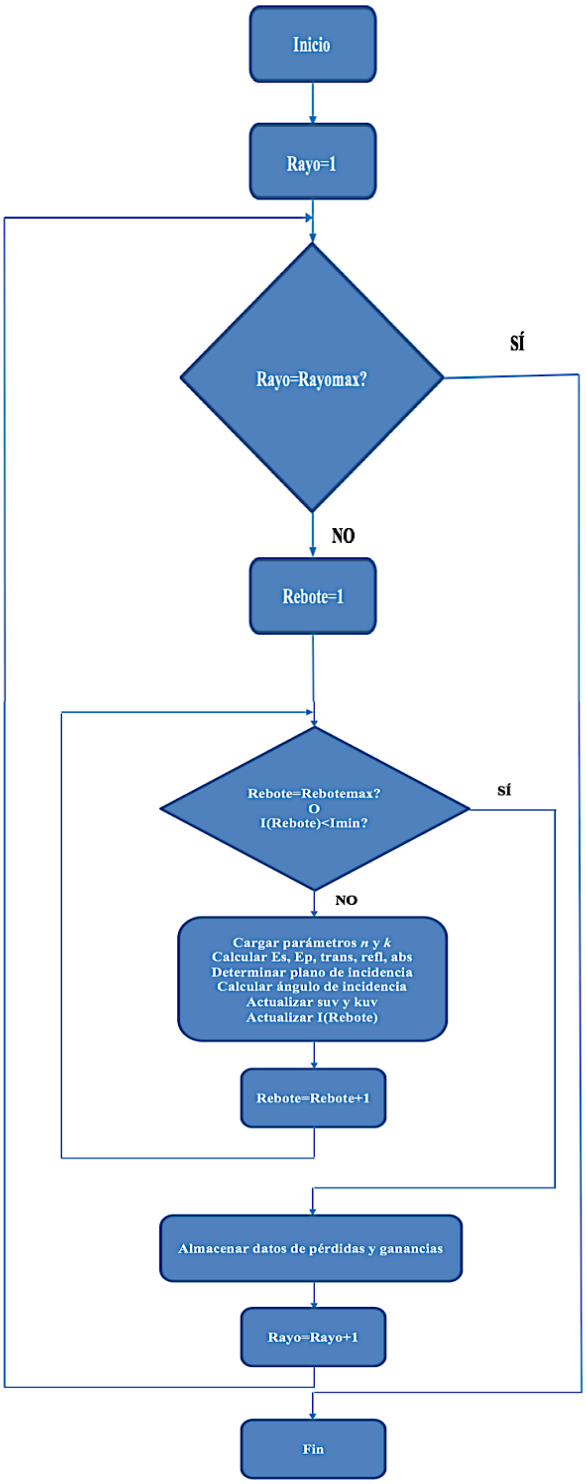

Diagrama de flujo 1. Estructura general de las simulaciones de rayos.

4.1.4 IQE y EQE

El valor de la eficiencia cuántica de una célula solar (QE, Quantum Efficiency) indica la cantidad de corriente que producirá la célula cuando sea irradiada por fotones de una determinada longitud de onda. Si la eficiencia cuántica de la célula está integrada en todo el espectro electromagnético

solar, como es el caso de este trabajo, se podrá evaluar la cantidad de corriente que producirá la célula cuando se expone a la luz solar. Se distingue entre eficiencia cuántica interna (IQE) y eficiencia cuántica externa (EQE).

Lo normal es conseguir primero el valor de EQE que se calcula a partir de la densidad de corriente espectral de cortocircuito total y el flujo espectral de fotones. Una vez se obtiene EQE, puede calcularse IQE como relación de EQE y 1-R siendo R el coeficiente de reflexión.

$$\text{EQE} = \frac{J_{sc\lambda}}{q \cdot \phi_o} \tag{10}$$

$$\text{IQE} = \frac{EQE}{1-R} \tag{11}$$

4.2 Simulaciones

Se van a realizar en total 3 simulaciones de Ray Tracing distintas modificando para cada una varios de los parámetros editables del código a fin de mostrar cuales pueden ser aquellos que más afecten a la eficiencia del módulo solar.

4.2.1 Simulación 1

Para esta primera simulación estos son los datos editables seleccionados:

Espectro	AM1.5g	**Películas**	0	**Anchura célula solar**	15,6 cm
Tipo de iluminación	Directa	**Capas**	2	**Radio célula solar**	10 cm
Número de Busbars	3	**Material capa 1**	EVA (McI09)	**Separación células X**	0,20 cm
Anchura de Busbars	0,150 cm	**Grosor capa 1**	0,200 cm	**Separación células Y**	0,20 cm
Longitud de fingers	0,200 cm	**Material capa 2**	Vidrio – Borosilicato (Schott BF33)	**Localización del rayo incidente**	Aleatorio
Anchura de fingers	150 μm	**Grosor capa 2**	0,045 cm	**Pérdidas en el borde del módulo**	Excluídas
Voc	0,630 V	**Material capa externa**	Aire	**Reflexión ARC celular**	Region 0
FF	0,785	**Longitud célula solar**	15,6 cm	**Máximo número de rayos**	50.000
Máximo número de rebotes por rayo	250	**Intensidad límite**	10^{-3} W/cm^2		

Tabla 2. Datos editables seleccionados para la primera simulación.

Esta primera simulación tuvo una duración total de 49,83 segundos. En ella se simularon 50.000 rayos que sumaron 239.952 rebotes en total. La distancia media recorrida por los rayos dentro del módulo fue de 0,7450 cm. La potencia total incidente al módulo fue de 22,5657 W que se dividió como sigue:

- 0,9184 W (4,07%) fueron reflejados al frente del módulo.
- 1,3635 W (6,04%) fueron transmitidos fuera del módulo.
- 0,5416 W (2,40%) fueron absorbidos por las capas.
- 0 W (0%) fueron absorbidos por las películas (no se incluyeron películas en esta simulación).
- 0,2539 W (1,13%) fueron absorbidos por la placa trasera metálica.
- 18,3057 W (81,12%) fueron absorbidos por las células solares.
- 1,1825 W (5,24%) fue la energía restante de los rayos que ya se consideró despreciable.

En el siguiente gráfico se observan estos resultados de forma más visual:

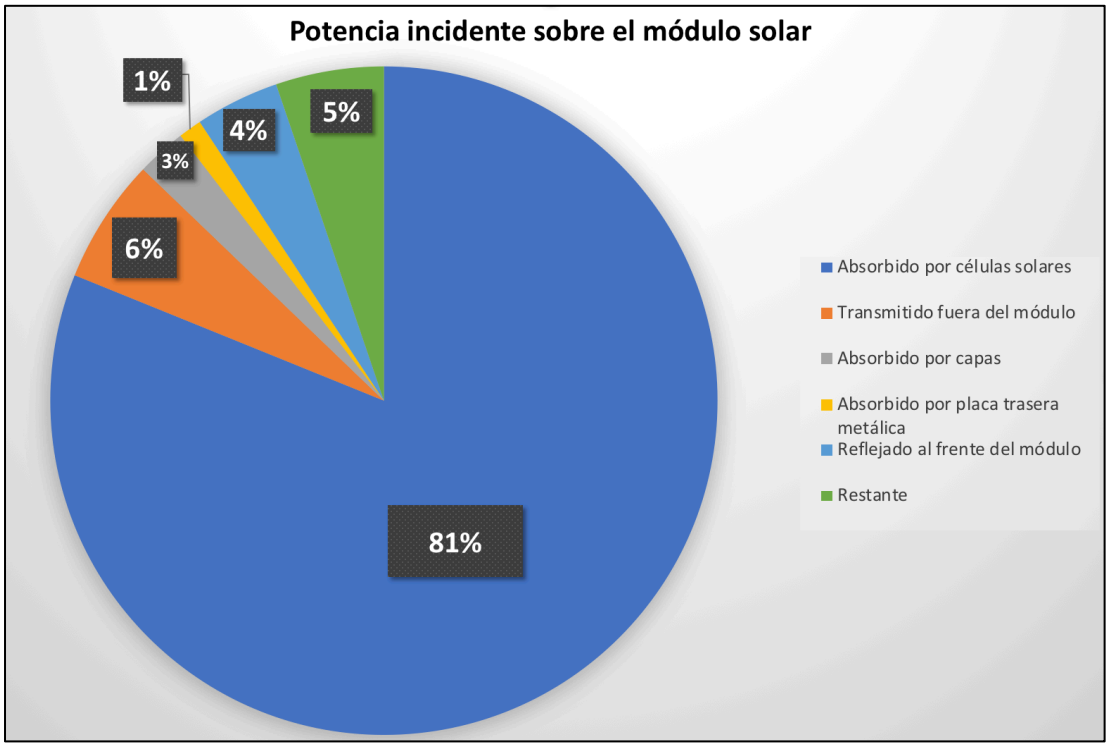

Figura 28. Potencia incidente sobre el módulo solar para la primera simulación.

La potencia final de salida del módulo solar se obtuvo con valor de 4,0782 W. Si se pone en relación esta potencia final de salida con la potencia total incidente del espectro sobre el módulo, se obtiene la eficiencia total de conversión del módulo que en esta simulación se sitúa en un 18,0725 %.

En cuanto a la eficiencia cuántica del módulo solar se comparan la interna y la externa en la siguiente gráfica:

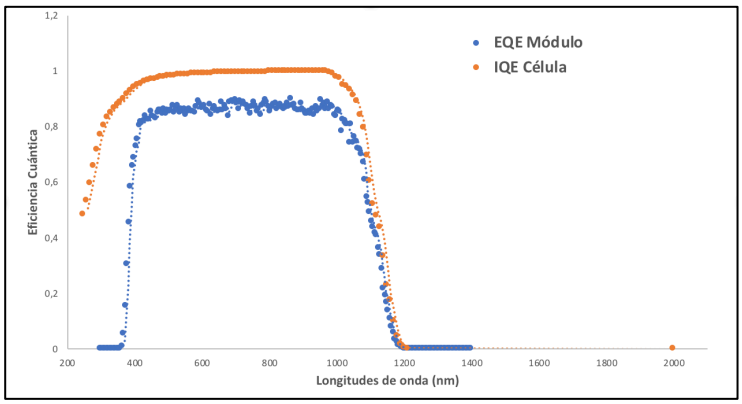

Figura 29. Eficiencia cuántica del módulo solar para la primera simulación.

Un IQE bajo indica que la capa activa de la célula solar no es capaz de hacer un buen uso de los fotones. La EQE por su parte depende tanto de la absorción como de la recolección de cargas. Un buen material evita la recombinación de carga, que causa una caída de la EQE. Una gráfica de eficiencia cuántica ideal tiene forma cuadrada, con el valor de eficiencia cuántica muy constante a lo largo de todo el espectro de estudio.

4.2.2 Simulación 2

Los datos de la segunda simulación son:

Espectro	12pm 06/12/1983 Buffalo, EEUU	**Películas**	0	**Anchura célula solar**	15,6 cm
Tipo de iluminación	Difusa	**Capas**	2	**Radio célula solar**	10 cm
Número de Busbars	3	**Material capa 1**	EVA (McI09)	**Separación células X**	0,20 cm
Anchura de Busbars	0,150 cm	**Grosor capa 1**	0,045 cm	**Separación células Y**	0,20 cm
Longitud de fingers	0,200 cm	**Material capa 2**	Vidrio – Borosilicato (Schott BF33)	**Localización del rayo incidente**	Aleatorio
Anchura de fingers	150 µm	**Grosor capa 2**	0,200 cm	**Pérdidas en el borde del módulo**	Excluídas
Voc	0,630 V	**Material capa externa**	Aire	**Reflexión ARC celular**	Region 0
FF	0,785	**Longitud célula solar**	15,6 cm	**Máximo número de rayos**	50.000
Máximo número de rebotes por rayo	250	**Intensidad límite**	10^{-3} W/cm^2		

Tabla 3. Datos editables seleccionados para la segunda simulación.

La duración de la simulación 2 fue de 50,9494 segundos. De nuevo se simularon 50.000 rayos que en este caso sumaron un total de 244.078 rebotes. La distancia media recorrida por cada uno de los rayos dentro del módulo fue de 0,7227 cm. La potencia total incidente en el módulo se redujo notablemente con respecto a la simulación 1 hasta el valor de 5,7221 W que en este estudio se dividió así:

- 0,2289 W (4,00%) fueron reflejados al frente del módulo.
- 0,3695 W (6,46%) fueron transmitidos fuera del módulo.
- 0,1437 W (2,51%) fueron absorbidos por las capas.
- 0 W (0%) fueron absorbidos por las películas (no se incluyeron películas en esta simulación).
- 0,072 W (1,26%) fueron absorbidos por la placa trasera metálica.
- 4,7754 W (83,46%) fueron absorbidos por las células solares.
- 0,1327 W (2,31%) fue la energía restante de los rayos que ya se consideró despreciable.

Representando estos resultados en un gráfico:

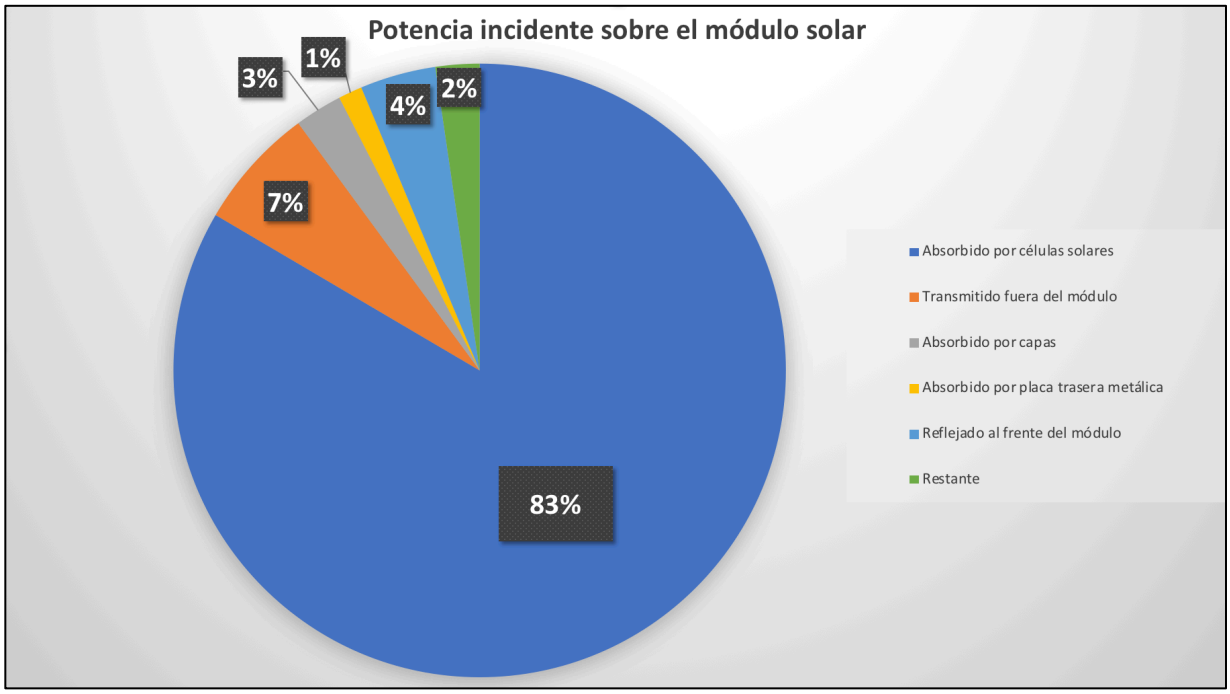

Figura 29. Potencia incidente sobre el módulo solar para la segunda simulación.

En este caso la potencia final de salida se redujo proporcionalmente con la potencia incidente total al módulo y tomó un valor de 1,0333 W por lo que la eficiencia del módulo se situó en un 18,0579 %.
La curva de la EQE en la gráfica de la eficiencia cuántica bajó un poco con respecto a la simulación 1 se muestra la Fig. 30.

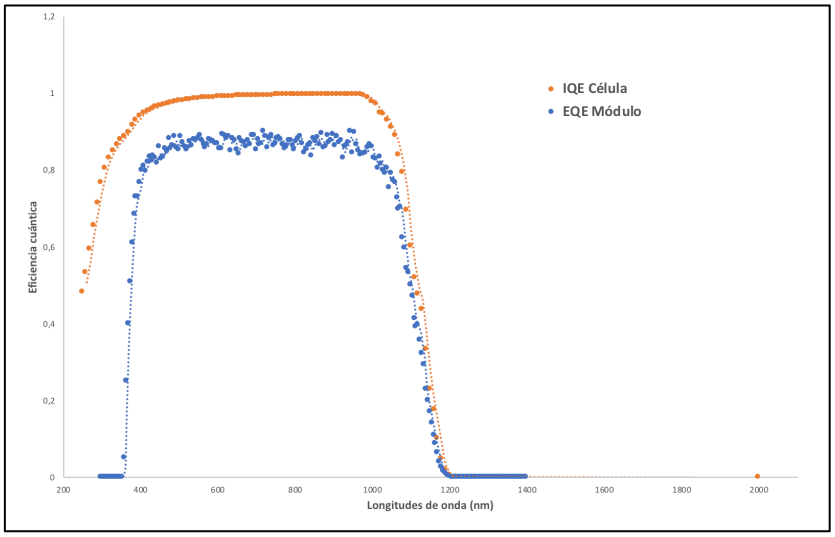

Figura 30. Eficiencia cuántica del módulo solar para la segunda simulación.

4.2.3 Simulación 3

En el caso de la tercera simulación, estos son los datos editables seleccionados:

Espectro	12pm 04/05/1967 Sacramento, EEUU	**Capas**	3	**Anchura célula solar**	15,6 cm
Tipo de iluminación	Directa	**Material capa 1**	EVA (McI09)	**Radio célula solar**	10 cm
Número de Busbars	3	**Grosor capa 1**	0,045 cm	**Separación células X**	0,20 cm
Anchura de Busbars	0,150 cm	**Material capa 2**	Vidrio – Borosilicato (Schott BF33)	**Separación células Y**	0,20 cm
Longitud de fingers	0,200 cm	**Grosor capa 2**	0,200 cm	**Localización del rayo incidente**	x=0,5 cm y=1,2 cm
Anchura de fingers	150 µm	**Material capa 3**	Silicio – DCC201 (McI09)	**Pérdidas en el borde del módulo**	Excluídas
Voc	0,630 V	**Grosor capa 3**	0,100 cm	**Reflexión ARC celular**	Region 0
FF	0,785	**Material capa externa**	Aire	**Máximo número de rayos**	50.000
Máximo número de rebotes por rayo	250	**Longitud célula solar**	15,6 cm		
Películas	0	**Intensidad límite**	10^{-3} W/cm^2		

Tabla 4. Datos editables seleccionados para la tercera simulación.

En el caso de la simulación 3, la duración fue de 112,021 segundos. Los 50.000 rayos simulados, triplicaron el número total de rebotes, con respecto a las dos simulaciones anteriores hasta 801.519. La distancia media recorrida por cada uno de los rayos también se incrementó notablemente siendo en este caso 3,3 centímetros. Sin embargo, en este caso, la potencia total incidente volvió a aumentar como consecuencia de la incidencia directa de los rayos sobre el módulo hasta los 13,2263 W que en esta ocasión se dividieron así:

- 0,6945 W (5,25%) fueron reflejados al frente del módulo.
- 4,7621 W (36,00%) fueron transmitidos fuera del módulo.
- 0,3725 W (2,82%) fueron absorbidos por las capas.
- 0 W (0%) fueron absorbidos por las películas (no se incluyeron películas en esta simulación).
- 2,4921 W (18,84%) fueron absorbidos por la placa trasera metálica.
- 3,8550 W (29,15%) fueron absorbidos por las células solares.
- 1,0502 W (7,94%) fue la energía restante de los rayos que ya se consideró despreciable.

Gráficamente se ve como en esta ocasión la potencia se reparte mucho más que en otras ocasiones:

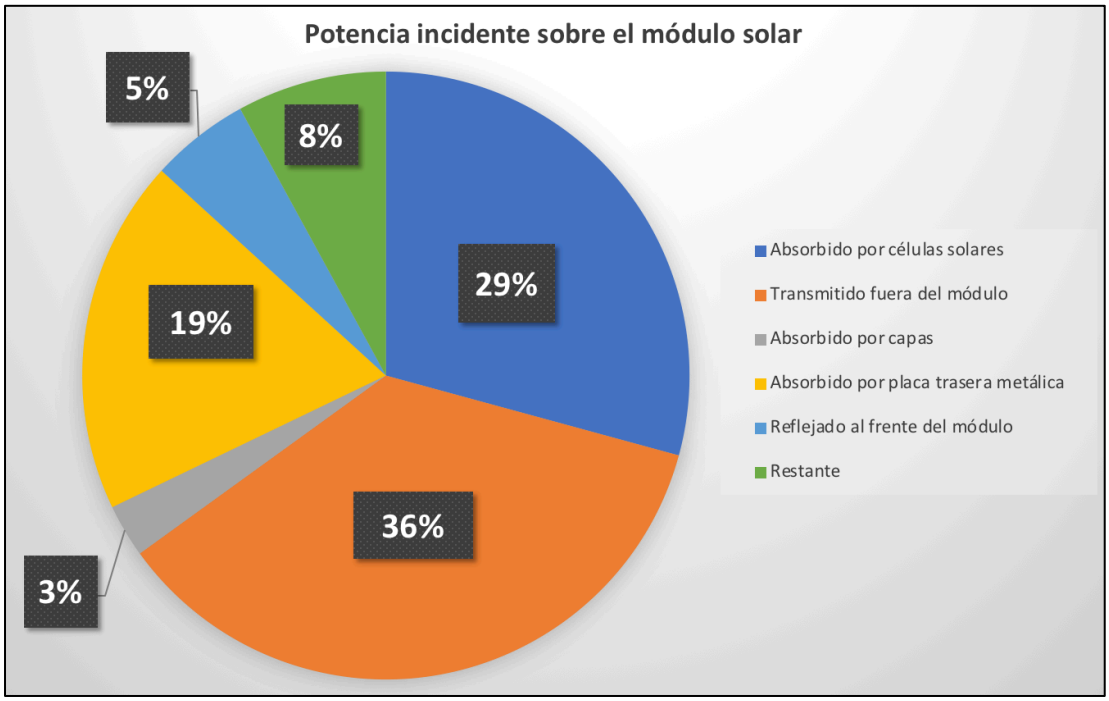

Figura 31. Potencia incidente sobre el módulo solar para la tercera simulación.

La potencia final de salida del módulo solar resultó 0,8731 W, lo que supuso un drástico descenso de la eficiencia del módulo hasta el 6,6031 %.

En este caso, la curva de EQE sí bajó ostensiblemente en la gráfica de la eficiencia cuántica con respecto a los otros dos ensayos:

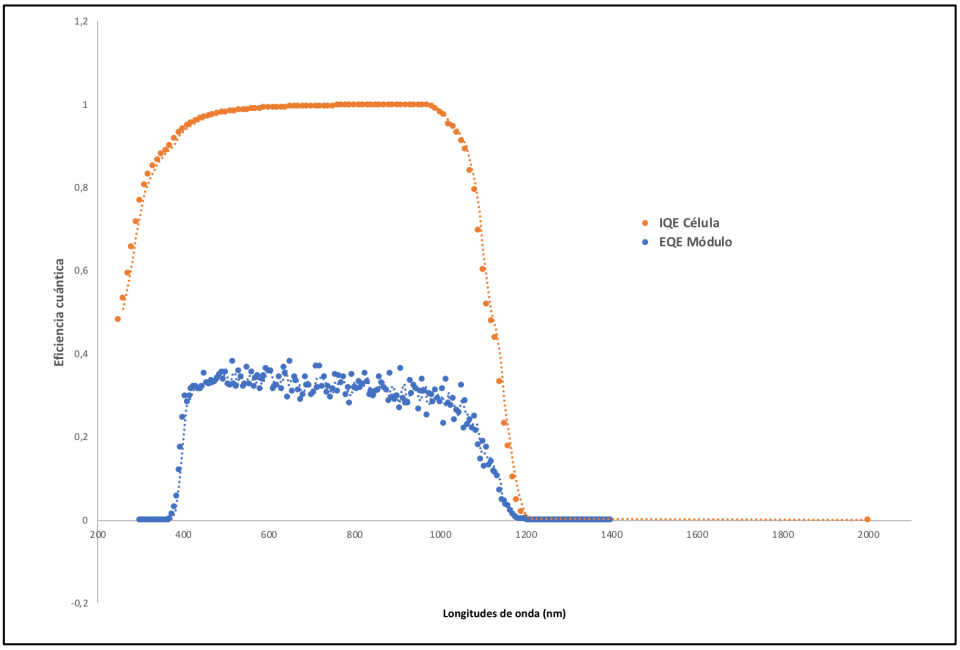

Figura 32. Eficiencia cuántica del módulo solar para la tercera simulación.

5 CONCLUSIONES

Una vez realizadas las simulaciones comentadas se obtienen una serie de conclusiones.

La principal diferencia entre el modelo de la **simulación 1** y el modelo de la **simulación 2** es que en el primero los rayos inciden de forma directa sobre el módulo mientras en el segundo lo hacen de manera difusa y aleatoria.

Esta incidencia no se traduce en una pérdida de eficiencia en porcentaje tal cual, pues se puede comprobar que en la primera se obtiene un rendimiento de conversión del 18,0725% mientras que en la segunda se consigue una conversión del 18,0579% de la energía total que llega al módulo.

Sin embargo, en términos absolutos, la conversión de energía solar en energía eléctrica se redujo hasta un 75% de la primera a la segunda simulación, consiguiendo un total de 4,0782 W y 1,0333 W.

Como conclusión, parece evidente que el ángulo de incidencia de los rayos que llegan al módulo de conversión de energía solar no tiene ningún efecto sobre el rendimiento del mismo, sino más bien sobre la cantidad de energía en términos absolutos que recoge el módulo. Esto es así porque no varía realmente la tecnología del módulo en sí, por lo que la eficiencia debe quedar inalterada, sino la cantidad de energía que recibe el propio módulo por la manera en que llegan a él los rayos que en estos dos casos concretos fueron 22,5657 W y 5,7221 W.

El número de rebotes totales de los 50.000 rayos simulados en cada experiencia también dan crédito a la hipótesis de que la diferencia de conversión total se debe exclusivamente a la forma en que llegan los rayos al módulo.

Para la tercera simulación se volvió a configurar como directa la incidencia de los rayos sobre el módulo solar. Sin embargo, se modificó el número de capas en el módulo, lo que afectó, esta vez sí al rendimiento de conversión con respecto a la **simulación 1**. En este caso la eficiencia del módulo se redujo hasta el 6,6031%.

Aumentaron drásticamente el porcentaje de pérdidas por ansorción de la placa trasera metálica y sobre todo la energía transmitida fuera del módulo. Tuvo, por tanto, un efecto bastante negativo la inclusión de una nueva capa en el diseño del módulo solar.

Por las características de este trabajo, no se han realizado más simulaciones que permitan obtener unas conclusiones más amplias que dotasen de una base sólida a las hipótesis formuladas. Sin embargo, queda claro que el programa desarrollado en MATLAB es versátil y permite introducir un gran número de variaciones para simular numerosas situaciones de diseño.

El programa tiene, por supuesto, mucho margen de mejora que por tiempo no se ha podido materializar. Se propone para una posible ampliación, la inclusión de un sistema de representación gráfica que permita vislumbrar los resultados de las simulaciones de una manera mucho más visual y dinámica. También se propone el uso de la biblioteca **OPTOMETRIKA** de MATLAB para el diseño y simulación también visual de los experimentos realizados por el programa que aquí se expone.

ANEXO

1. Códigos de MATLAB

1.1 Código completo

```
%%                              C"DIGO COMPLETO
%%

%%                    INICIALIZACI"N DE VARIABLES(1-559)
%%

tic;

SumDistancia=0;
SumIPerdidoReflexExt=0;
SumIPerdidoAbsPorCapa(1)=0;
SumIPerdidoAbsPorCapa(2)=0;
SumIPerdidoAbsPorCapa(3)=0;
SumIPerdidoAbsPorCapa(4)=0;
SumIPerdidoAbsPorCapa(5)=0;
SumIPerdidoAbsPorCapa(6)=0;
SumIPerdidoAbsPorCapaTotal=0;
SumIPerdidoAbsPorPelicula(1)=0;
SumIPerdidoAbsPorPelicula(2)=0;
SumIPerdidoAbsPorPelicula(3)=0;
SumIPerdidoAbsPorPelicula(4)=0;
SumIPerdidoAbsPorPelicula(5)=0;
SumIPerdidoAbsPorPeliculaTotal=0;
SumIPerdidoAbsPlano(1)=0;
SumIPerdidoAbsPlano(2)=0;
SumIPerdidoAbsPlano(3)=0;
SumIPerdidoAbsPlano(4)=0;
SumIPerdidoAbsPlano(5)=0;
SumIPerdidoAbsPlano(6)=0;
SumIPerdidoAbsPlanoTotal=0;
SumIPerdidoTransPlano(1)=0;
SumIPerdidoTransPlano(2)=0;
SumIPerdidoTransPlano(3)=0;
SumIPerdidoTransPlano(4)=0;
SumIPerdidoTransPlano(5)=0;
SumIPerdidoTransPlano(6)=0;
SumIPerdidoTransPlanoTotal=0;
SumIGanadoAbsPlano(1)=0;
SumIGanadoAbsPlano(2)=0;
SumIGanadoAbsPlano(3)=0;
SumIGanadoAbsPlano(4)=0;
SumIGanadoAbsPlano(5)=0;
SumIGanadoAbsPlano(6)=0;
SumIGanadoAbsPlanoTotal=0;
SumIRestante=0;
```

```
InteraccionCelula=0;
cnt=0;
cntab=0;
cntb=0;

    % InicializaciÛn de algunos par·metros con Bin ya determinado %

GananciaCelulaSolarPorLongitudOndaInicial=zeros(2,221);      % 221 es el
Bin m·s grande posible cuando Pvalue valiera 1. Si no hacemos esto el
programa intentar· acceder a Ìndices que no existen y parar· su ejecuciÛn
con este error.
GananciaCelulaSolarPorLongitudOndaFinal=zeros(2,221);
RayosPorLongitudOndaInicial=zeros(1,221);
PerdidasPorLongitudOndaInicial=zeros(19,221);
PerdidasJscPorLongitudOndaInicial=zeros(9,221);

    % Fin inicializaciÛn de algunos par·metros con Bin ya determinado %

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% DEFINIR AQUÕ LAS VARIABLES
QUE EN EL EXCEL SE DEFINEN CON CELDAS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%% CELDAS DE ADDITIONAL INPUTS %%%%

MaxNumFilasEscritas=1000;
NombreEspectro='1967May04 12pm Sacramento (Mar99)';
TipoIluminacion='Directa'; % Las opciones son difusa o directa
theta0=180; % ¡ngulo cenital (sexagesimal)
phi0=270; % ¡ngulo acimutal (sexagesimal)
FactorEscalaIntensidad=1;
IntensidadEspectral=0.100000;    % W/(cm^2)

%%%%% CELDAS MAIN PAGE %%%%%

%%% SUPERFICIE FRONTAL METALIZADA %%%

NumeroBusBars=3;
AnchuraBusBars=0.150;       % cm
LongitudFingers=0.200;      % cm
AnchuraFingers=0.015;       % 150 micrometros=0.015 cm

%%% FIN SUPERFICIE FRONTAL METALIZADA %%%

%%% DIMENSIONES DE LA C…LULA %%%

xCelula=15.6;         % cm
yCelula=15.6;         % cm
rCelula=10;           % cm
xSep=0.20/2;          % cm
ySep=0.20/2;          % cm
if rCelula>sqrt((xCelula/2)^2+(yCelula/2)^2)
    AreaCelula=xCelula*yCelula;
else
    AreaCelula=4*(0.5*(xCelula/2)*sqrt(rCelula^2-
(xCelula/2)^2)+0.5*(yCelula/2)*sqrt(rCelula^2-
(yCelula/2)^2)+pi*(rCelula^2)*(((pi/2)-acos(0.5*xCelula/rCelula)-
acos(0.5*yCelula/rCelula))/(2*pi)));
end
```

```
AreaSobrante=(xCelula+xSep)*(yCelula+ySep)-AreaCelula;

%%% FIN DIMENSIONES DE LA C…LULA %%%

%%% PAR¡METROS EL…CTRICOS %%%

Voc=0.630;      % V (Voltaje en circuito abierto)
FF=0.785;       % Factor de llenado (fill factor), adimensional. Potencia
m·xima de la cÈlula solar entre el producto del voltaje en circuito
abierto (Voc) y la corriente de cortocircuito (Isc)

%%% FIN PAR¡METROS EL…CTRICOS %%%

%%% GROSOR Y MATERIALES DE LAS CAPAS %%%

CapaG(1)=0;          % cm. CÈlula. En el excel no hay celda para rellenar.
CapaG(2)=0.045;      % cm. En excel esta serÌa la capa 1 (layer 1), pero
en MATLAB los subÌndices de los vectores empiezan en 0, por lo que el
vector que en MATLAB tomamos como capa(1), serÌa la propia cÈlula
fotovoltaica. MATERIAL: EVA (Mcl09).
CapaG(3)=0.200;      % cm. MATERIAL: Glass-borosilicate (Schott EF33)
CapaG(4)=0.100;          % cm.
CapaG(5)=0;          % cm.
CapaG(6)=0;          % cm. Externa. En el excel no hay celda para
rellenar.

NombreMaterialCapa{1}='None';
NombreMaterialCapa{2}='EVA (McI09)';
NombreMaterialCapa{3}='Glass - borosilicate (Schott BF33)';
NombreMaterialCapa{4}='Silicone - DCC201 (McI09)';
NombreMaterialCapa{5}='None';
NombreMaterialCapa{6}='Air';

%%% FIN GROSOR Y MATERIALES DE LAS CAPAS %%%

%%% GROSOR Y MATERIALES DE PELÕCULAS SOBRE CAPAS %%%

PeliculaG(1)=0;         % cm--> 1 cm=10^7 nm
PeliculaG(2)=0;         % cm
PeliculaG(3)=0;         % cm
PeliculaG(4)=0;         % cm
PeliculaG(5)=0;         % cm

PeliculaG(6)=0;     % Prevenir error en lÌnea 176

NombreMaterialPelicula{1}='Not applicable';
NombreMaterialPelicula{2}='None';
NombreMaterialPelicula{3}='None';
NombreMaterialPelicula{4}='None';
NombreMaterialPelicula{5}='None';

NombreMaterialPelicula{6}='None';   % Prevenir error en lÌnea 176

%%% FIN GROSOR Y MATERIALES DE PELÕCULAS SOBRE CAPAS %%%

%%%%% FIN CELDAS MAIN PAGE %%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
% DefiniciÛn de constantes

qq=1.602e-19;
hh=6.626e-34;
cc=299790000;       % Velocidad de la luz

% N˙mero m·ximo de filas en ficheros RI
DatoMax=2000;

% N˙mero de regiones
RegionMax=4;

% Cargar informaciÛn espectral
if strcmp(NombreEspectro,'Single wavelength')==0
    [PEspectro, FicheroDeIntensidadDeEspectro,
PrimeraLongitudDeOndaDeEspectro,
UltimaLongitudDeOndaDeEspectro]=LoadSpectrum(NombreEspectro);
end

% Cargar iluminaciÛn incidente (define el ·ngulo de incidencia)
if  strcmp(TipoIluminacion,'Directa')
    IluminacionIncidenteEsIsotropica='TRUE'; % TRUE
    theta0=pi/2+asin(rand)+0.000001;    % No se si esto es necesario aquÌ,
la intuiciÛn me dice que sÌ
    phi0=2*pi*rand;                      % No se si esto es necesario
aquÌ, la intuiciÛn me dice que sÌ
else
    IluminacionIncidenteEsIsotropica='FALSE'; % FALSE
    theta0=theta0*pi/180;       % Radianes
    phi0=phi0*pi/180;           % Radianes
    if theta0<=pi/2 || theta0>pi
        disp('Theta tiene que estar entre 90 y 180 grados')
        break
    end
end

if isnumeric(FactorEscalaIntensidad)==1
    if FactorEscalaIntensidad<=0
        FactorEscalaIntensidad=0.0001;
    end
    if FactorEscalaIntensidad>=100000
        FactorEscalaIntensidad=100000;
    end
else
    FactorEscalaIntensidad=1;
end

% C·lculo de la intensidad incidente, con el factor de escala y el
cos(theta0)
if strcmp(NombreEspectro,'Single wavelength')==0

Iinc=abs(FicheroDeIntensidadDeEspectro*FactorEscalaIntensidad*cos(theta0))
;
else
    Iinc=abs(IntensidadEspectral*FactorEscalaIntensidad*cos(theta0));
```

```
end

% Intensidad del rayo inicial
Iinicial=1;

% Cargar opciones
TratoBordeModulo='Excluido (modulo infinito)';      % La otra opciÛn es
'incluido (modulo singular)'
CalculoReflexionARC='Region 0';                     % La otra opciÛn es
'planar & especular'

% Cargar informaciÛn bin para espectro de salida
LongitudDeOndaInicialBin=300;     % nm
LongitudDeOndaFinalBin=1400;      % nm
BinGrosor=5;                  % nm
Bins=(LongitudDeOndaFinalBin-LongitudDeOndaInicialBin)/BinGrosor;

% TransmisiÛn dependiente de la cantidad de Bins
TransmisionPlanoPorLongitudOndaFinal=zeros(7,Bins+1);

% Cargar valores de convergencia
bmax=250;       % Rebotes m·ximos por rayo
rayomax=50000;  % N˙mero m·ximo de rayos
% if InitialSheet='Output by bounce'
% rayomax=1;
% end
Imin=10^-3;
ActualizarPantallaCada=1000;        % 1000 rayos en este caso

% Cargar datos n, k, t de capas y pelÌculas

CapaColumn=10;              % Columna J de la p·gina principal
EspesorCapaColumn=12;       % Columna L de la p·gina principal
PeliculaColumn=13;          % Columna M de la p·gina principal
EspesorPeliculaColumn=16;   % Columna P de la p·gina principal
PrimeraFilaMaterial=5;

MaxNumCapasUsuario=5;       % M·ximo n˙mero de capas de usuario (1=cÈlula,
MaxNumCapaUsuario=external)
MaxNumPeliculasUsuario=MaxNumCapasUsuario-1;

% ReestructuraciÛn de capas para desechar capas intermedias vacÌas y
cargar n, k & t para capas y pelÌculas
capa=1;
for FilaCapa=(PrimeraFilaMaterial+MaxNumCapasUsuario):-
1:PrimeraFilaMaterial

    CapaPresente='True';
    if strcmp(NombreMaterialCapa{11-FilaCapa},'None') ||
strcmp(NombreMaterialCapa{11-FilaCapa},'____') ||
strcmp(NombreMaterialCapa{11-FilaCapa},'Not applicable') || CapaG(11-
FilaCapa)<=0
        CapaPresente='False';
    end
    PeliculaPresente='True';
    if strcmp(NombreMaterialPelicula{11-FilaCapa},'None') ||
strcmp(NombreMaterialPelicula{11-FilaCapa},'____') ||
strcmp(NombreMaterialPelicula{11-FilaCapa},'Not applicable') ||
PeliculaG(11-FilaCapa)<=0
```

```
        PeliculaPresente='False';
    end

    switch FilaCapa
        case PrimeraFilaMaterial+MaxNumCapasUsuario             % Capa
cÈlula (capa 0 y pelicula 0)
            switch CalculoReflexionARC
                case 'Region 0'                                 % Lectura
de superficies para todas las regiones
                    NombreMaterialCapa{11-FilaCapa}='Not applicable';
                    NombreMaterialPelicula{11-FilaCapa}='Not applicable';
                    PeliculaG(11-FilaCapa)=0;
                    CapaG(11-FilaCapa)=0;
                case 'planar & specular'                        %
Establecer Region 0 a especular, leer en el tipo de superficie para otras
dos regiones
                    if strcmp(NombreCapaMaterial{11-FilaCapa},'None') ||
strcmp(NombreCapaMaterial{11-FilaCapa},'____') ||
strcmp(NombreCapaMaterial{11-FilaCapa},'Not applicable')      % Prevenir
la celda en blanco de la capa
                        NombreCapaMaterial{11-FilaCapa}='Si - crystalline
(Gre08)';
                    end
                    [CapaMaterialn{11-FilaCapa},CapaMaterialk{11-
FilaCapa}]=LoadRIdata(NombreCapaMaterial{11-FilaCapa},capa);
                    if strcmp(PeliculaPresente,'False')
                        NombreMaterialPelicula{11-FilaCapa}='None';
                        PeliculaG(11-FilaCapa)=0;
                    else
                        [PeliculaMaterialn{11-
FilaCapa},PeliculaMaterialk{11-
FilaCapa}]=LoadRIdata(NombreMaterialPelicula{11-FilaCapa}, capa);
                        PeliculaG(11-FilaCapa);
                    end
            end
            capa=capa+1;

        case MaxNumCapasUsuario                 % Capa externa (sin
pelicula)
            if strcmp(NombreMaterialCapa{11-MaxNumCapasUsuario},'None') ||
strcmp(NombreMaterialCapa{11-MaxNumCapasUsuario},'____')              %
Prevenir la celda en blanco de la capa
                NombreMaterialCapa{11-MaxNumCapasUsuario}='Air';
            end
            [CapaMaterialn{11-FilaCapa},CapaMaterialk{11-
FilaCapa}]=LoadRIdata(NombreMaterialCapa{11-FilaCapa}, capa);
            CapaG(capa)=0;

        otherwise       % Otras capas
%             NombreMaterialCapa{11-FilaCapa}='None';       % AquÌ no
sirven por la forma en que se ha programado este cÛdigo, los nombres est·n
arriba definidos "por defecto"
%             NombreMaterial{11-FilaCapa}='None';
%             Pelicula(11-FilaCapa)=0;
%             CapaG(11-FilaCapa)=0;
            if strcmp(CapaPresente,'True')
                [CapaMaterialn{11-FilaCapa},CapaMaterialk{11-
FilaCapa}]=LoadRIdata(NombreMaterialCapa{11-FilaCapa},capa);
```

```
                  CapaG(11-FilaCapa);
                    if strcmp(PeliculaPresente,'True')
                        [PeliculaMaterialn{11-
FilaCapa},PeliculaMaterialk{11-
FilaCapa}]=LoadRIdata(NombreMaterialPelicula{11-FilaCapa}, capa);
                        PeliculaG(11-FilaCapa);
                    else
                        PeliculaG(capa)=0;
                    end
              capa=capa+1;
              else
                  % No hay iteraciÛn
              end
    end
end

%%%%%%%%%%%%%%%% Arreglando CapaMaterialn y CapaMaterialk %%%%%%%%%%%%%%%%

if sum(CapaG>0)

tamano=length(CapaMaterialn);
for i=1:tamano
    if strcmp(class(CapaMaterialn{i}),'cell')
        tamano1=length(CapaMaterialn{i});
        for j=1:tamano1
            if isempty(CapaMaterialn{i}{j})==0
               CapaMaterialn{i}=CapaMaterialn{i}{j};
               break
            end
        end
    end
end
tamano=length(CapaMaterialk);
for i=1:tamano
    if strcmp(class(CapaMaterialk{i}),'cell')
        tamano1=length(CapaMaterialk{i});
        for j=1:tamano1
            if isempty(CapaMaterialk{i}{j})==0
               CapaMaterialk{i}=CapaMaterialk{i}{j};
               break
            end
        end
    end
end

%%%%%% Poner CapaMaterialn y CapaMaterialk con la dimensiÛn adecuada
%%%%%%

j=1;
for i=1:tamano
    if isempty(CapaMaterialn{i})==0
        CapaMaterialnAux{j}=CapaMaterialn{i};
        j=j+1;
    end
end
CapaMaterialn=CapaMaterialnAux;

j=1;
for i=1:tamano
```

```
    if isempty(CapaMaterialk{i})==0
        CapaMaterialkAux{j}=CapaMaterialk{i};
        j=j+1;
    end
end
CapaMaterialk=CapaMaterialkAux;

end

%%%% Fin poner CapaMaterialn y CapaMaterialk con la dimensiÛn adecuada
%%%%

%%%%%%%%%%%%%% Fin Arreglando CapaMaterialn y CapaMaterialk %%%%%%%%%%%%%%

%%%%%%%%%%%%%% Arreglando PeliculaMaterialn y PeliculaMaterialk
%%%%%%%%%%%%%%
if sum(PeliculaG>0)
tamano=length(PeliculaMaterialn);
for i=1:tamano
    if strcmp(class(PeliculaMaterialn{i}),'cell')
        tamano1=length(PeliculaMaterialn{i});
        for j=1:tamano1
            if isempty(PeliculaMaterialn{i}{j})==0
               PeliculaMaterialn{i}=PeliculaMaterialn{i}{j};
               break
            end
        end
    end
end
tamano=length(PeliculaMaterialk);
for i=1:tamano
    if strcmp(class(PeliculaMaterialk{i}),'cell')
        tamano1=length(PeliculaMaterialk{i});
        for j=1:tamano1
            if isempty(PeliculaMaterialk{i}{j})==0
               PeliculaMaterialk{i}=PeliculaMaterialk{i}{j};
               break
            end
        end
    end
end

% Poner PeliculaMaterialn y PeliculaMaterialk con la dimensiÛn adecuada %

% j=1;
% for i=1:tamano
%     if isempty(PeliculaMaterialn{i})==0
%         PeliculaMaterialnAux{j}=PeliculaMaterialn{i};
%         j=j+1;
%     end
% end
% PeliculaMaterialn=PeliculaMaterialnAux;
%
% j=1;
% for i=1:tamano
%     if isempty(PeliculaMaterialk{i})==0
%         PeliculaMaterialkAux{j}=PeliculaMaterialk{i};
%         j=j+1;
```

```
%     end
% end
% PeliculaMaterialk=PeliculaMaterialkAux;

% Fin poner PeliculaMaterialn y PeliculaMaterialk con la dimensiÛn
adecuada %

end

%%%%%%%%%% Fin Arreglando PeliculaMaterialn y PeliculaMaterialk %%%%%%%%%%

% Si despuÈs de este bucle, capa sigue siendo igual a 1 significa que el
usuario no ha definido ninguna capa.

if sum(CapaG)==0
    disp('La simulaciÛn requiere que se defina al menos 1 capa (Capa 1),
con un material y un espesor.');
    disp('Escoja "Aire" como material para las capas Externa y Capa 1 si
quiere simular una cÈlula encapsulada.');
    break
end

capamax=capa;               % N˙mero de capas elegido.
peliculamax=capamax-1;      % N˙mero de pelÌculas elegido.

%%%%%%%%%%%%% Cargar datos de reflexiÛn en el plano 1 (plano bajo
independiente del ·ngulo de incidencia) %%%%%%%%%%%%%

% La primera columna de cada regiÛn corresponde a las longitudes de onda
(lambda) expresadas en nm. La segunda columna son las reflexiones
correspondientes a cada longitud de onda de cada una de las regiones.

    Reflexion{1}=load('Reflexion en region 0.txt');     % ARC cÈlula
    Reflexion{2}=load('Reflexion en region 1.txt');     % Backsheet
    Reflexion{3}=load('Reflexion en region 2.txt');     % Metal

%%%%%%%%%%%%% Fin cargar datos de reflexiÛn en el plano 1 (plano bajo
independiente del ·ngulo de incidencia) %%%%%%%%%%%%%



% En la segunda columna conviene no tener elementos nulos por posibles
problemas en c·lculos futuros.

    for i=1:3
        [M,N]=size(Reflexion{i});
        for j=1:M
            if Reflexion{i}(j,2)==0
                Reflexion{i}(j,2)=0.0000000001;
            end
        end
    end

    switch CalculoReflexionARC
        case 'Region 0'                 % Lectura en el tipo de
superficie para todas las regiones.
        case 'planar & especular'       % Poner region 0 a especular,
lectura en el tipo de superficie para dos regiones m·s.
```

```
            [FraccionEspecular0]=1;      % Pone la primera componente a
1?????
        otherwise
            disp('Problema encontrado leyendo el tipo de superficie para
el plano 1, Region 0');
            break
    end

    % for region=1:RegionMax            % No es neceario el for para el
programa en MATLAB
        % La segunda coordenada serÌa la de la regiÛn. La traducciÛn se
basa en que el visual basic usa el cÛdigo ASCII para la primera
coordenada. Por ejemplo para la region 0(1) da 70 que es el cÛdigo ASCII
de la letra F.
        FraccionEspecular(1,1)=1;           % 100 por ciento
        FraccionEspecular(1,2)=0;           % 0 por ciento
        FraccionEspecular(1,3)=0;           % 0 por ciento
    % end
    for p=2:(6+capamax)
        FraccionEspecular(p,1)=1;           % Se fuerza la reflexion
especular para el resto de planos
    end

    % Cargar dimensiones de la cÈlula
    % Call LoadCellDimensions(xCell, yCell, rCell, xSep, ySep, geomFlag) %
No hace falta en MATLAB, ya est·n cargadas arriba lÌneas (26-40).

    % Dimensiones de la busbar convertidas a cm
    % Call CheckValidation("Main Page", "F4", 2, 0, 1000, True) % No hace
falta en MATLAB, ya est·n cargadas arriba lÌneas (17-24).

    % TODO IGUAL HASTA FILA 446

    % C·lculo de las dimensiones unitarias de la cÈlula
    xUnidad=xCelula+2*xSep;         % Asume muchas cÈlulas infinitas, para
que la cÈlula unidad tenga 0.5*xSep en cada parte de la cÈlula
    yUnidad=yCelula+2*ySep;

    zCapa(1)=0;         % Altura de la cÈlula/backsheet (necesidad de
mantenerla a cero o vacÌa)

    for capa=2:capamax
        zCapa(capa)=zCapa(capa-1)+CapaG(capa);
    end
    zUnidad=zCapa(capamax);         % Suma de todas las capas excluyendo
cÈlula (capa 0) y externa (capamax)

    % Vectores unitarios de los planos
        [nuv]=SetPlaneUnitVector;       % Esto podrÌa hacerlo arriba en
lugar de tener una funciÛn especÌfica para ello. M·s conveniente?

    % Cargar localizaciÛn del rayo
    RayoLocalizacion='Localizacion singular';          % La otra opciÛn es
'Localizacion singular'

    % Cambiar y utilizar en caso de que RayoLocalizacion='LocalizaciÛn
singular'
    x0=0.5;     % cm
```

```
    y0=1.2;     % cm

    if x0<=0
        x0=0.00000001;
    end
    if x0>=xUnidad
        x0=xUnidad-0.00000001;
    end
    if y0<=0
        y0=0.00000001;
    end
    if y0>=yUnidad
        y0=yUnidad-0.00000001;
    end

    % Cargar datos IQE de las cÈlulas solares
    IQECelula=load('IQE en areas activas de la celula solar (region
0).txt');

    % InicializaciÛn de c·lculos de desviaciones est·ndar
    IndiceAlmacen=0;            % Seguramente tenga que cambiarlo a 1. No
hace falta, suma 1 a la hora de usarlo, no hay problema de Ìndice=0.
    AlmacenSumIntervalos=1;
    ContadorCalcPpal=0;         % Sirve para escribir datos en las hojas
excel, quiz·s in˙til en este MATLAB.

%%          BEGIN RAY TRACING/COMINEZO DEL RAY TRACING(560-577)
%%

for rayo=1:rayomax

%%      PREPARE FOR NEW RAY/PREPARACI”N PARA UN NUEVO RAYO(578-644)
%%

% Pongo las variables a cero.

IPerdidoReflexExt=0; %I se refiere a energÌa?
%peliculamax=3;
for capa=2:capamax-1 % se ignoran la capa 1 (cÈlula) y la capa 6 (externa)
    IPerdidoAbsPorCapa(capa)=0;
end
IPerdidoAbsPorCapaTotal=0;

for pelicula=1:peliculamax
    IPerdidoAbsPorPelicula(pelicula)=0;
end
IPerdidoAbsPorPeliculaTotal=0;

for p=1:6 % seis planos externos para contabilizar pÈrdidas por
transmitancia y absorbancia
    IPerdidoTransPlano(p)=0;
    IPerdidoAbsPlano(p)=0;
    IGanadoAbsPlano(p)=0;
end
IPerdidoTransPlanoTotal=0;
IPerdidoAbsPlanoTotal=0;
% IPerdidoAbsPorBacksheet=0;
% IPerdidoAbsPorMetal(0)=0;
% IPerdidoAbsPorMetal(1)=0;
```

```
% IPerdidoAbsPorMetal(2)=0;
IGanadoAbsPlanoTotal=0;
IRestante=0;
DistanciaDeRayo=0;
region=0;

if strcmp(NombreEspectro,'Single wavelength')
    w=650; % nm
else
    Pvalue=rand;
    w=InterpolateData(PEspectro,2,Pvalue,0,0,1,2);
end

Bin=DetermineWavelengthBin(w,LongitudDeOndaInicialBin,LongitudDeOndaFinalB
in,BinGrosor,Bins);

IQECelulaInterpolado=InterpolateData(IQECelula,2,w,0,0,1,2);

capan=zeros(1,capamax);
capak=zeros(1,capamax);

for capa=1:capamax-1      % En el excel pone capa=1:capamax pero hay que
tener en cuenta que nuestra capa 1 es la cÈlula, en VB es la 0.
    if isempty(CapaMaterialn{capa})==0
        capan(capa+1)=InterpolateData(CapaMaterialn,3,w,capa,0,1,2);
        capak(capa+1)=InterpolateData(CapaMaterialk,3,w,capa,0,1,2);
        capaAlfa(capa+1)=capak(capa+1)*4*pi/(0.0000001*w);
    end
end

for pelicula=1:peliculamax     % pelicula=1:peliculamax.Idem que con
capas.
    if PeliculaG(pelicula)>0


peliculan(pelicula)=InterpolateData(PeliculaMaterialn,3,w,pelicula,0,1,2);

peliculak(pelicula)=InterpolateData(PeliculaMaterialk,3,w,pelicula,0,1,2);
    end
end

capa=capamax-1;
while CapaG(capa)<=0
    capa=capa-1;
end
pelicula=capa;

%%                INCIDENT RAY/RAYO INCIDENTE(644-666)                %%

thetaInc=pi-theta0;
%Por si quiero verlos en grados sexagesimales
theta0G=theta0*180/pi;
phiG=phi0*180/pi;
thetaIncG=thetaInc*180/pi;
%if rayo==rayomax
    %call
    %writebouncedata(-1,-
1,"","","","","","","","","","","","","","",IStart,
```

```
    %w)
%end

%%       FIRST BOUNCE (EXTERNAL)/PRIMER REBOTE (EXTERNO)(667-721)       %%

% Todos los Ìndices que eran 0 en VB son 1 en MATLAB, todos los que eran
1, son 2 en MATLAB
b=1;    %Evitamos Ìndices problem·ticos poniendo b=1 en lugar de b=0.

% seleccionar localizaciÛn inicial

p=2; %capa entre la m·s alta y la externa
pos(3,b)=zUnidad; % configurar zUnit

if strcmp(RayoLocalizacion,'Localizacion singular')
     % "Single location"
        pos(1,b)=x0;
        pos(2,b)=y0;
elseif strcmp(RayoLocalizacion,'Aleatorio')
     % "Random"
        pos(1,b)=rand*xUnidad;
        pos(2,b)=rand*yUnidad;
end
rayoX0=pos(1,b); % valores iniciales guardados en "Output by Ray"
rayoY0=pos(2,b);

% establecimiento del vector unitario inicial s (perpendicular al plano de
incidencia)

suv(1,1)=sin(phi0);
suv(2,1)=-cos(phi0);
suv(3,1)=0;

% GeneraciÛn del ·ngulo de polarizaciÛn para el rayo incidente al plano
externo y rebote 0 b=0
% Luz no polarizada de momento

xi=rand*2*pi;
E(1,1,b)=cos(xi); % Componente real de Es
E(1,2,b)=0;       % Componente imaginaria de Es
E(2,1,b)=sin(xi); % Componente real de Ep
E(2,2,b)=0;       % Componente imaginaria de Ep

% c·lculo de la reflexiÛn, transmitancia y absorbancia para el primer
rebote b=0
% primero ponemos los datos de RI (??) en el formato adecuado
if PeliculaG(capa)>0 % Si la capa tiene una pelÌcula sobre su superficie
orientada hacia arriba
    RI(1,1)=capan(capamax);  % n de la capa externa % En VB ponia capamax
    RI(1,2)=capak(capamax);  % k de la capa externa (irrelevante si se
convierte en cero) % En VB ponÌa capamax
    RI(2,1)=peliculan(pelicula); % n de la capa m·s externa
    RI(2,2)=peliculak(pelicula) ;% k de la capa m·s externa
    RI(3,1)=capan(capa);  % n de la capa m·s externa
    RI(3,2)=capak(capa);  % k de la capa m·s externa
else
    RI(1,1)=capan(capamax);  % n de la capa externa % En VB ponÌa capamax
    RI(1,2)=capak(capamax);  % k de la capa externa (irrelevante si se
convierte en cero) % En VB ponÌa capamax
```

```
    RI(2,1)=capan(capa);  % n de la capa m·s externa
    RI(2,2)=capak(capa);  % k de la capa m·s externa
end
% calcular Es, Ep, R, A y T

[Erefl,Etrans,refl,trans,absorb]=NewCalculateRAT(b,PeliculaG(pelicula),RI,
thetaInc,w,Iinicial,E);
% C·lculo de pÈrdidas

IPerdidoReflexExt=Iinicial*refl;
IPerdidoAbsorbPorPelicula(pelicula)=Iinicial*absorb;  % AbsorciÛn en la
pelÌcula m·s externa

%%    FIRST RAY WITHIN MODULE/PRIMER RAYO DENTRO DEL M”DULO (722-758)
%%

% Tomar Es & Ep como rayo existente del rebote 1. b=1.

E(1,1,b)=Etrans(1,1);
E(1,2,b)=Etrans(1,2);
E(2,1,b)=Etrans(2,1);
E(2,2,b)=Etrans(2,2);

% Seguimiento del rayo transmitido

I(b)=Iinicial*trans;

% C·lculo del ·ngulo de transmisiÛn

thetaSalida=asin(capan(capamax)*sin(thetaInc)/capan(capa));      % QU…
HAGO CON CAPAMAX????

% ConversiÛn del ·ngulo de tranmisiÛn a theta (·ngulo de incidencia)

theta=pi-thetaSalida;

% C·lculo del nuevo vector unitario para el rayo 1 (es decir, el rayo
despuÈs del rebote 1 y antes del rebote 2)

kuv(1,b)=sin(theta)*cos(phi0);
kuv(2,b)=sin(theta)*sin(phi0);
kuv(3,b)=cos(theta);

% C·lculo del vector unitario s para el rebote existente b=0
% Como el rayo es transmitido, s se da la vuelta, es decir, rota por pi

suv(1,1)=suv(1,1);
suv(2,1)=-suv(2,1);
% suv(3,0)=0;       % No varÌa por no variar la direcciÛn, sÛlo el
sentido.

% Escribe los datos del rebote 0 (plano y posiciÛn), y rayo 0
(direcciÛn,intensidad, longitud de onda).
if rayo==rayomax % Escribe los datos del rayo final
    % Call
writebouncedata(b,p,region,pos(1,b),pos(2,b),pos(3,b),kuv(1,b),kuv(2,b),ku
v(3,b),suv(1,b),suv(2,b),suv(3,b),E(0,0,b),E(0,1,b),E(1,0,b),E(1,1,b),I(b)
,w)
```

```
end
%%  SUBSEQUENT BOUNCES AND RAYS/REBOTES Y RAYOS SUBSECUENTES(759-1462)  %%

% Bucle que analiza todos los rebotes, en este script el principio, pero
no cierro el bucle for. (759-782)

for b=2:bmax
    % La metalizaciÛn resetea las normales al plano, necesitamos
redefinirlos en cada iteraciÛn. El nuv para el plano equivalente 1 serÌa 0
0 1 de nuevo
    nuv(1,1)=0;
    nuv(1,2)=0;
    nuv(1,3)=1;
    cntb=cntb+1;
    % DeterminaciÛn del bin de salida de la longitud de onda final

Binfinal=DetermineWavelengthBin(w,LongitudDeOndaInicialBin,LongitudDeOndaF
inalBin,BinGrosor,Bins);   % No estoy seguro de quÈ deberÌa poner en lugar
de Bins

    % Caso Bin=0 y Binfinal=0 %

    if Bin==0
        Bin=1;
    end
    if Binfinal==0
        Binfinal=1;
    end

    % Fin caso Bin=0 y Binfinal=0 %

    IPerdidoAbsPorCapa=zeros(MaxNumCapasUsuario);
    IPerdidoAbsPorPelicula=zeros(MaxNumPeliculasUsuario);

    % Para que el rayo no sea paralelo a los planos x,y o z

    if kuv(1,b-1)==0
        kuv(1,b-1)=0.0000000001;
    end
    if kuv(2,b-1)==0
        kuv(2,b-1)=0.0000000001;
    end
    if kuv(3,b-1)==0
        kuv(3,b-1)=0.0000000001;
    end

    %%   FIND NEW POSITION X,Y,Z/DETERMINAR NUEVA POSICI”N X,Y,Z(783-836)
%%

% DeterminaciÛn del plano de interseccIÛn m·s cercano y el punto de
interseccIÛn

for p=1:6
    dotproduct=kuv(1,b-1)*nuv(p,1)+kuv(2,b-1)*nuv(p,2)+kuv(3,b-
1)*nuv(p,3);

    if dotproduct<0     % El rayo ha interceptado un plano
        % DeterminaciÛn de x,y,z de la interseccIÛn. Salida del bucle
cuando se encuentre el nuevo P.
```

```
        switch p
            case 1      % Parte de abajo de la capa
                pos(3,b)=zCapa(capa-1);      % Rayo viajando bajo de
manera que z est· en la capa baja
                c1=(pos(3,b)-pos(3,b-1))/kuv(3,b-1);
                pos(1,b)=pos(1,b-1)+c1*kuv(1,b-1);
                pos(2,b)=pos(2,b-1)+c1*kuv(2,b-1);

                if pos(1,b)>=0 && pos(1,b)<=xUnidad && pos(2,b)>=0 &&
pos(2,b)<=yUnidad
                    if pos(3,b)==0
                        p=1;
                    else
                        p=6+(capa-2);
                    end
                    break       % No estoy seguro de si este break va aquÌ
o dentro del else
                end

            case 2      % Parte de arriba de la capa
                    pos(3,b)=zCapa(capa);       % Rayo viajando alto  de
manera que z est· en la capa alta
                    c1=(pos(3,b)-pos(3,b-1))/kuv(3,b-1);
                    pos(1,b)=pos(1,b-1)+c1*kuv(1,b-1);
                    pos(2,b)=pos(2,b-1)+c1*kuv(2,b-1);
                    if pos(1,b)>=0 && pos(1,b)<=xUnidad && pos(2,b)>=0 &&
pos(2,b)<=yUnidad
                        if pos(3,b)==zUnidad
                            p=2;
                        else
                            p=6+capa-1;
                        end
                        break   % No estoy seguro de si este break va aquÌ
o dentro del else
                    end
            case {3,4}        % Sur, Norte
                if p==3
                    pos(2,b)=0;
                else
                    pos(2,b)=yUnidad;
                end
                c1=(pos(2,b)-pos(2,b-1))/kuv(2,b-1);
                pos(1,b)=pos(1,b-1)+c1*kuv(1,b-1);
                pos(3,b)=pos(3,b-1)+c1*kuv(3,b-1);
                if pos(1,b)>=0 && pos(1,b)<=xUnidad && pos(3,b)>=0 &&
pos(3,b)<=zUnidad
                    break
                end
            case {5,6}        % Este, Oeste
                if p==5
                    pos(1,b)=0;
                end
                if p==6
                    pos(1,b)=xUnidad;
                end
                c1=(pos(1,b)-pos(1,b-1))/kuv(1,b-1);
                pos(2,b)=pos(2,b-1)+c1*kuv(2,b-1);
```

```
                pos(3,b)=pos(3,b-1)+c1*kuv(3,b-1);
                if pos(2,b)>=0 && pos(2,b)<=yUnidad && pos(3,b)>=0 &&
pos(3,b)<=zUnidad
                    break
                end
        end
    end
end

% C·lculo de la distancia al plano m·s cercano

Distancia=sqrt((pos(1,b)-pos(1,b-1))^2+(pos(2,b)-pos(2,b-1))^2+(pos(3,b)-
pos(3,b-1))^2);

%% IF BOTTOM PLANE, CALCULATE WHETHER RAY INTERSECTS WITH INNER (0), OUTER
(1), OR METAL (2) REGION OF PLANE/SI ES EL PLANO BAJO, C¡LCULO DE LA
INTERSECCI”N CON LAS REGIONES DEL PLANO INTERNA (0), EXTERNA (1), O METAL
(2)(837-885)

if p==1     % Chequeo del plano bajo
    if pos(1,b)<xSep || pos(2,b)>(xSep+xCelula)     % Chequeo de X
        region=1;   % Externo
    else            % Chequeo de Y
        if pos(2,b)<ySep || pos(2,b)>(ySep+yCelula)
            region=1;   % Externo
        else        % Chequeo r
            if sqrt((pos(1,b)-xSep-xCelula/2)^2+(pos(2,b)-ySep-
yCelula/2)^2)>rCelula
                region=1;
            else
                region=0;
            end
        end
    end
else
    region=0;
end

% Si el rayo intersecta la region 0 (cÈlula solar), determina si el rayo
reflecta la metalizaciÛn (region 2)

if p==1 && region==0
    if NumeroBusBars>0 && AnchuraBusBars>0      % Busbar son las barras
"verticales" de la placa
        LongFinger=(xCelula-
NumeroBusBars*AnchuraBusBars)/(2*NumeroBusBars);      % Finger son las
barritas "horizontales" de la placa. FingerLength=pitchFinger?? NO.
        for c1=0:NumeroBusBars-1
            BordeIzq=c1*(2* LongFinger+AnchuraBusBars)+ LongFinger;
            BordeDer=BordeIzq+AnchuraBusBars;
            if pos(1,b)>=BordeIzq && pos(1,b)<=BordeDer
                region=2;
            end
        end
    end

    % Determina si el ray reflecta en el finger
    if LongitudFingers>0
```

```
        LocalizacionyUnidad=pos(2,b)-
LongitudFingers*(floor(((double(1000000)*pos(2,b))/(double(1000000)*Longit
udFingers)))); % Barra invertida--> Equivalente a floor, redondeo por
truncamiento
        if LocalizacionyUnidad<0
            LocalizacionyUnidad=LocalizacionyUnidad+LongitudFingers;
        end
        if LocalizacionyUnidad>=0 && LocalizacionyUnidad<=AnchuraFingers
            region=2;
        end
    end
end

% DeterminaciÛn del plano equivalente de interseccciÛn

if p<=6
    PlanoEquivalente=p;
else
    if kuv(3,b-1)>0
        PlanoEquivalente=2;
    else
        PlanoEquivalente=1;
    end
end

% Aumenta la distancia del rayo

DistanciaDeRayo=DistanciaDeRayo+Distancia;

%% ACCOUNT FOR HOST ABSORPTION LOSS/CUENTA DE P…RDIDA POR ABSORBANCIA DEL
HOST(886-906) %%

TransmisionHost=exp(-capaAlfa(capa)*Distancia);
if TransmisionHost>0
    RaizCTransmisionHost=sqrt(TransmisionHost);
else
    RaizCTransmisionHost=0;
end

% Recuento de la pÈrdida de absorciÛn y determinaciÛn de E en el rebote b

E(1,1,b)=E(1,1,b-1)*RaizCTransmisionHost;
E(1,2,b)=E(1,2,b-1)*RaizCTransmisionHost;
E(2,1,b)=E(2,1,b-1)*RaizCTransmisionHost;
E(2,2,b)=E(2,2,b-1)*RaizCTransmisionHost;

I(b)=I(b-1)*TransmisionHost;
IAbsPorCapa(capa)=I(b-1)-I(b);
IPerdidoAbsPorCapa(capa)=IPerdidoAbsPorCapa(capa)+IAbsPorCapa(capa);

if I(b)>=Imin           % Se cierra justo antes de
Write_bounce_data_if_final_ray

    % Entiendo que cuando I es menor que Imin, ya no merece la pena seguir
teniendo encuenta su valor.
    % Si I(b)=0 el programa petarÌa calculando la reflexiÛn y transmisiÛn.
```

```
    %% ROTATE E AXIS (s,k,p) INTO NEW PLANE OF INCIDENCE/ROTACI”N DEL EJE
DE E CONFORME A UN NUEVO PLANO DE INCIDENCIA(907-959) %%

% Rota E desde la salida del rebote b-1 hasta la entrada al rebote b
% C·lculo de la matriz de rotaciÛn [cosBeta,-sinBeta;sinBeta,cosBeta]
% ComprobaciÛn de que el rayo es perpendicular al nuevo plano, es decir
dotproduct=-1

if dotproduct<-0.9999
    suv(1,b)=-suv(1,b-1);
    suv(2,b)=-suv(2,b-1);
    suv(3,b)=-suv(3,b-1);
    cosBeta=-1;
else
    % C·lculo del vector suv del rayo incidente al plano nuv en el rebote
b
    [suv,kuv(1,b),kuv(2,b),kuv(3,b)]=CalculateNewSuv(b,kuv(1,b-1),kuv(2,b-
1),kuv(3,b-1),dotproduct,nuv,PlanoEquivalente);
    cosBeta=suv(1,b-1)*suv(1,b)+suv(2,b-1)*suv(2,b)+suv(3,b-1)*suv(3,b);
end

if cosBeta<-0.999       % beta=pi, E rota 180∫
    E(1,1,b)=-E(1,1,b);
    E(1,2,b)=-E(1,2,b);
    E(2,1,b)=-E(2,1,b);
    E(2,2,b)=-E(2,2,b);

elseif cosBeta>0.999
    E(1,1,b)=-E(1,1,b);
    E(1,2,b)=-E(1,2,b);
    E(2,1,b)=-E(2,1,b);
    E(2,2,b)=-E(2,2,b);

elseif cosBeta<1
    sinBeta=sqrt(1-cosBeta^2);
    EsNuevaRe=cosBeta*E(1,1,b)-sinBeta*E(2,1,b);
    EsNuevaIm=cosBeta*E(1,2,b)-sinBeta*E(2,2,b);
    EpNuevaRe=sinBeta*E(1,1,b)+cosBeta*E(2,1,b);
    EpNuevaIm=sinBeta*E(1,2,b)+cosBeta*E(2,2,b);
    E(1,1,b)=EsNuevaRe;
    E(1,2,b)=EsNuevaIm;
    E(2,1,b)=EpNuevaRe;
    E(2,2,b)=EpNuevaIm;

else        % No deberÌa ocurrir nunca, significarÌa que el vector del
rayo es el mismo que en la iteraciÛn anterior. IMPOSIBLE.
    disp('Error. Vector S invariado desde la ˙ltima interacciÛn')
    % Exit Sub
end

%% CALCULATE INTERACTION LOSS (RAT), GAIN AND NEW INTENSITY/C¡LCULO DE LAS
P…RDIDAS, GANANCIAS Y NUEVA INTENSIDAD DE LA INTERACCI”N(960-1161) %%

% C·lculo del ·ngulo de incidencia theta
theta=acos(-dotproduct);

% Encuentra ninc y nout que se usar·n para los c·lculos de la reflexiÛn
aquÌ y para los c·lculos de la refracciÛn m·s adelante
```

```
ninc=capan(capa);
kinc=capak(capa);

switch p
    case {3,4,5,6}        % paredes
        nout=capan(capamax);        % nout es el RI externo (kout es
irrelevante).  % De nuevo la paradoja de capamax.???
    otherwise           % fondo y superior
        if kuv(3,b-1)>0
            nout=capan(capa+1);
            kout=capak(capa+1);
            pelicula=capa;      % La direcciÛn del rayo es hacia arriba,
asÌ que se aproxima a la pelÌcula en la parte superior de la capa
        else
            nout=capan(capa-1);
            kout=capak(capa-1);
            pelicula=capa-1;    % La direcciÛn del rayo es hacia abajo,
asÌ que se aproxima a la pelÌcula en la parte inferior de la capa
        end
end

% C·lculo de la reflexiÛn, absorciÛn y transmisiÛn
switch p
    case 1      % Fondo
        % Ganancia: absorbido por la cÈlula; refl: reflejado; absorb:
absorbido por la pelicula, backsheet o metal; trans: transmitido fuera del
plano 1 (cero)
        if region==0        % El rayo intersecta la cÈlula ARC
            switch CalculoReflexionARC
                case 'Region 0'     % NO LO ENTIENDO BIEN
                    [refl]=InterpolateData(Reflexion,3,w,region+1,0,1,2);
% El 'region+1' lo pongo para evitar Ìndice=0 dentro de InterpolateData.
No pongo directamente region=1 para preservar el nombre de
'CalculoReflARC'.
                    ganancia=1-refl;
                    trans=0;
                    Erefl(1,1)=sqrt(refl)*E(1,1,b);
                    Erefl(1,2)=sqrt(refl)*E(1,2,b);
                    Erefl(2,1)=sqrt(refl)*E(2,1,b);
                    Erefl(2,2)=sqrt(refl)*E(2,2,b);
                case 'Calculated (planar & specular)'
                    if PeliculaG(1)>0
                        RI(1,1)=ninc;           % n de la capa m·s externa
                        RI(1,2)=kinc;           % k de la capa m·s externa
(aunque es irrelevante)
                        RI(2,1)=peliculan(1);   % n de la pelÌcula en la
cÈlula
                        RI(2,2)=peliculak(1);   % k de la pelÌcula en la
cÈlula
                        RI(3,1)=nout;           % n de la cÈlula
                        RI(3,2)=kout;           % k de la cÈlula
                    else
                        RI(1,1)=ninc;           % n de la capa m·s externa
                        RI(1,2)=kinc;           % k de la capa m·s externa
(aunque es irrelevante)
                        RI(2,1)=nout;           % n de la cÈlula
                        RI(2,2)=kout;           % k de la cÈlula
                    end
```

```
[Erefl,Etrans,refl,trans,absorb]=NewCalculateRAT(b,PeliculaG(pelicula),RI,
theta,w,I(b),E);
            end
        else
            [refl]=InterpolateData(Reflexion,3,w,region+1,0,1,2);
            % Region puede ser una busbar, finger1 o finger2
            ganancia=0;
            trans=0;
            absorb=1-refl;

            % AsignaciÛn del campo elÈctrico Erefl
            Erefl(1,1)=sqrt(refl)*E(1,1,b);
            Erefl(1,2)=sqrt(refl)*E(1,2,b);
            Erefl(2,1)=sqrt(refl)*E(2,1,b);
            Erefl(2,2)=sqrt(refl)*E(2,2,b);
        end
    case {3,4,5,6}        % Paredes
        switch TratoBordeModulo
            case 'Incluido (modulo individual)'
                RI(1,1)=ninc;           % n de la capa m·s externa
                RI(1,2)=kinc;           % k de la capa m·s externa (aunque
es irrelevante)
                RI(2,1)=nout;           % n de la cÈlula
                RI(2,2)=kout;           % k de la cÈlula

[Erefl,Etrans,refl,trans,absorb]=NewCalculateRAT(b,PeliculaG(pelicula),RI,
theta,w,I(b),E);
            case 'Excluido (modulo infinito)'
                refl=1;
                trans=0;
                absorb=0;
                % El rayo continua como si no hubiera reflexiÛn por lo que
E se mantiene igual
                Erefl(1,1)=E(1,1,b);
                Erefl(1,2)=E(1,2,b);
                Erefl(2,1)=E(2,1,b);
                Erefl(2,2)=E(2,2,b);
        end
    otherwise       % Top
        if PeliculaG(pelicula)>0
            RI(1,1)=ninc;           % n de la capa m·s externa
            RI(1,2)=kinc;           % k de la capa m·s externa (aunque es
irrelevante)
            RI(2,1)=peliculan(pelicula);   % n de la pelÌcula en la cÈlula
            RI(2,2)=peliculak(pelicula);   % k de la pelÌcula en la cÈlula
            RI(3,1)=nout;           % n de la cÈlula
            RI(3,2)=kout;           % k de la cÈlula
        else
            RI(1,1)=ninc;           % n de la capa m·s externa
            RI(1,2)=kinc;           % k de la capa m·s externa (aunque es
irrelevante)
            RI(2,1)=nout;           % n de la cÈlula
            RI(2,2)=kout;           % k de la cÈlula
        end


[Erefl,Etrans,refl,trans,absorb]=NewCalculateRAT(b,PeliculaG(pelicula),RI,
theta,w,I(b),E);
```

```
end

% Si el plano est· compuesto de cÈlulas solares, calcula ganancias y
escribe algunos datos
if p==1
    switch region
        case 0
            InteraccionCelula=InteraccionCelula+1;
            IGanadoCelula=I(b)*ganancia;
            IGanadoCelulaIQE=I(b)*ganancia*IQECelulaInterpolado;
            % Introducir datos en Bin arrays

GananciaCelulaSolarPorLongitudOndaInicial(1,Bin)=GananciaCelulaSolarPorLon
gitudOndaInicial(1,Bin)+I(b)*ganancia;

GananciaCelulaSolarPorLongitudOndaInicial(2,Bin)=GananciaCelulaSolarPorLon
gitudOndaInicial(2,Bin)+I(b)*ganancia*IQECelulaInterpolado*qq*(0.000000001
*w)/(hh*cc); % Corriente incidente

GananciaCelulaSolarPorLongitudOndaFinal(1,Binfinal)=GananciaCelulaSolarPor
LongitudOndaFinal(1,Binfinal)+I(b)*ganancia;

GananciaCelulaSolarPorLongitudOndaFinal(2,Binfinal)=GananciaCelulaSolarPor
LongitudOndaFinal(2,Binfinal)+I(b)*ganancia*IQECelulaInterpolado*qq*(0.000
000001*w)/(hh*cc); % Corriente incidente

        case 1
            % IPerdidoAbsPorBacksheet=IPerdidoAbsPorBacksheet+I(b)*absorb;
        case 2
            % IPerdidoAbsPorMetal(0)=IPerdidoAbsPorMetal(0)+I(b)*absorb;
        case 3
            % IPerdidoAbsPorMetal(1)=IPerdidoAbsPorMetal(1)+I(b)*absorb;
        case 4
            % IPerdidoAbsPorMetal(2)=IPerdidoAbsPorMetal(2)+I(b)*absorb;
    end
end

% C·lculo de las pÈrdidas y las ganancias y la nueva intensidad
% PÈrdida transmisiÛn primero

if p<=6
    IPerdidoTransPlano(p)=IPerdidoTransPlano(p)+I(b)*trans;

TransmisionPlanoPorLongitudOndaFinal(p,Binfinal)=TransmisionPlanoPorLongit
udOndaFinal(p,Binfinal)+I(b)*trans;     % trans=0 para plano 1
end

% PÈrdida y ganancia absorb
if p==1
    if region==0        % CÈlulas solares
        IGanadoAbsPlano(p)=IGanadoAbsPlano(p)+I(b)*ganancia;
        IPerdidoAbsPorPelicula(1)=IPerdidoAbsPorPelicula(1)+I(b)*absorb;
% PÈrdida atribuida por absorciÛn a la pelÌcula fina (pelÌcula 1)
    else
        IPerdidoAbsPlano(p)=IPerdidoAbsPlano(p)+I(b)*absorb;        %
PÈrdida atribuida por absorciÛn al backsheet/metal
    end
else
```

```
    if p==2 || p>=7

IPerdidoAbsPorPelicula(pelicula)=IPerdidoAbsPorPelicula(pelicula)+I(b)*abs
orb;        % PÈrdida atribuida por absorciÛn a la pelÌcula fina
    else
        if absorb>0
            % disp('ATENCI”N: AbsorciÛn atribuida al plano lateral')
            cnt=cnt+1;
            cntab=cntab+absorb;
        end
    end
end
% C·lculo de la nueva intensidad
if p<=6
    I(b)=I(b)*refl;
else
    % Si la interacciÛn es con una capa interna, entonces sÛlo se reduce
la intensidad por absorciÛn
    I(b)=I(b)*(1-absorb);
end

% Para superficies internas, antes de decidir si seguir el rayo reflejado
o el transmitido ajustamos Erefl y Etrans.
if p>6
    if absorb<1
        divisor=sqrt(1-absorb);
        Erefl(1,1)=Erefl(1,1)/divisor;      % Devuelve la magnitud al
nivel anterior
        Erefl(1,2)=Erefl(1,2)/divisor;
        Erefl(2,1)=Erefl(2,1)/divisor;
        Erefl(2,2)=Erefl(2,2)/divisor;
        Etrans(1,1)=Etrans(1,1)/divisor;
        Etrans(1,2)=Etrans(1,2)/divisor;
        Etrans(2,1)=Etrans(2,1)/divisor;
        Etrans(2,2)=Etrans(2,2)/divisor;
    else
        Erefl(1,1)=0;
        Erefl(1,2)=0;
        Erefl(2,1)=0;
        Erefl(2,2)=0;
        Etrans(1,1)=0;
        Etrans(1,2)=0;
        Etrans(2,1)=0;
        Etrans(2,2)=0;
    end
end

%% CALCULATE NEW RAY UNIT VECTOR k AND s/C¡LCULO DEL NUEVO VECTOR UNITARIO
k Y s(1162-1339) %%

% DeterminaciÛn del tipo de rayo
switch p
    case {1,2,3,4,5,6}        % Interfaces externas
        if rand<FraccionEspecular(p,region+1)       % 'region+1' para
evitar Ìndice=0.
            TipoRadiacionEmitida='Especular';
        else
            TipoRadiacionEmitida='Lambertiana';
        end
```

```
    otherwise
        if rand<refl/(refl+trans)       % Seguir rayo reflejado
            TipoRadiacionEmitida='Especular';
        else                            % Seguir rayo transmitido
            TipoRadiacionEmitida='Refraccion';
            if kuv(3,b-1)>0
                capa=capa+1;
            else
                capa=capa-1;
            end
        end
end

switch TipoRadiacionEmitida
    case 'Lambertiana'
        % Calculamos primero el vector para enfocar de cara el plano (i.e.
plano con normal (0,0,1))
        theta_lamb=asin(sqrt(rand));
        phi_lamb=rand*2*pi;
        kuv_lamb(1)=sin(theta_lamb)*cos(phi_lamb);
        kuv_lamb(2)=sin(theta_lamb)*sin(phi_lamb);
        kuv_lamb(3)=cos(theta_lamb);

        if region>=2        % En VB pone region>=2, pero por coherencia
con lo que hemos estado haciendo con region, pongo 3. Al final estamos
trabajando con region desde 0 asÌ que vuelvo a poner 2, si falla lo
analizaremos.
            % RotaciÛn para distribuir alrededor de la superficie local
normal. La componente t es normal al plano, u y v son perpendiculares a
esta direcciÛn.
            % La componente u satisface ux*tx+uy*ty+uz*tz=0. Una opciÛn es
u=[-tz 0 tx]. De manera que el vector v es paralelo al producto vectorial
de t y u.
            % v=[ty*uz-tz*uy tz*ux-tx*uz tx*uy-ty*ux].
            t(1)=nuv(p,1);
            t(2)=nuv(p,2);
            t(3)=nuv(p,3);
            u(1)=-1*t(3);
            u(2)=0;
            u(3)=t(1)+t(2);     % La adiciÛn de t(2) evita el error de
t(1)=0
            LongitudU=sqrt(u(1)^2+u(2)^2+u(3)^2);
            u(1)=u(1)/LongitudU;
            u(2)=u(2)/LongitudU;
            u(3)=u(3)/LongitudU;
            v(1)=t(2)*u(3)-t(3)*u(2);
            v(2)=t(3)*u(1)-t(1)*u(3);
            v(3)=t(1)*u(2)-t(2)*u(1);
            LongitudV=sqrt(v(1)^2+v(2)^2+v(3)^2);
            v(1)=v(1)/LongitudV;
            v(2)=v(2)/LongitudV;
            v(3)=v(3)/LongitudV;
            % RotaciÛn de kuv_lamb dentro de la nueva base (kuv_lamb(3)
debe estar en la direcciÛn de la normal)

kuv_lamb(1)=kuv_lamb(1)*u(1)+kuv_lamb(2)*v(1)+kuv_lamb(3)*t(1);

kuv_lamb(2)=kuv_lamb(1)*u(2)+kuv_lamb(2)*v(2)+kuv_lamb(3)*t(2);
```

```
kuv_lamb(3)=kuv_lamb(1)*u(3)+kuv_lamb(2)*v(3)+kuv_lamb(3)*t(3);
            LongitudK=sqrt(kuv_lamb(1)^2+kuv_lamb(2)^2+kuv_lamb(3)^2);
            kuv_lamb(1)=kuv_lamb(1)/LongitudK;
            kuv_lamb(2)=kuv_lamb(2)/LongitudK;
            kuv_lamb(3)=kuv_lamb(3)/LongitudK;
        end

        % Nuevo vector suv. Calculado como si el rayo acabase de
reflejarse en el plano 1.
        denom_lamb=sqrt(kuv_lamb(1)^2+kuv_lamb(2)^2);
        suv_lamb(1)=kuv_lamb(2)/denom_lamb;
        suv_lamb(2)=-kuv_lamb(1)/denom_lamb;
        suv_lamb(3)=0;

        switch PlanoEquivalente
            case 1      % No hay rotaciÛn
                kuv(1,b)=kuv_lamb(1);
                kuv(2,b)=kuv_lamb(2);
                kuv(3,b)=kuv_lamb(3);
                suv(1,b)=suv_lamb(1);
                suv(2,b)=suv_lamb(2);
                suv(3,b)=suv_lamb(3);
            case 2      % RotaciÛn de 180ª entorno al eje y
                kuv(1,b)=-kuv_lamb(1);
                kuv(2,b)=kuv_lamb(2);
                kuv(3,b)=-kuv_lamb(3);
                suv(1,b)=-suv_lamb(1);
                suv(2,b)=suv_lamb(2);
                suv(3,b)=-suv_lamb(3);
            case 3      % RotaciÛn de 90ª entorno al eje x
                kuv(1,b)=kuv_lamb(1);
                kuv(2,b)=kuv_lamb(3);
                kuv(3,b)=-kuv_lamb(2);
                suv(1,b)=suv_lamb(1);
                suv(2,b)=suv_lamb(3);
                suv(3,b)=-suv_lamb(2);
            case 4      % RotaciÛn de -90™ entorno al eje x

                kuv(1,b)=kuv_lamb(1);
                kuv(2,b)=-kuv_lamb(3);
                kuv(3,b)=kuv_lamb(2);
                suv(1,b)=suv_lamb(1);
                suv(2,b)=-suv_lamb(3);
                suv(3,b)=suv_lamb(2);
            case 5      % RotaciÛn de 90ª entorno al eje y

                kuv(1,b)=kuv_lamb(3);
                kuv(2,b)=kuv_lamb(2);
                kuv(3,b)=-kuv_lamb(1);
                suv(1,b)=suv_lamb(3);
                suv(2,b)=suv_lamb(2);
                suv(3,b)=-suv_lamb(1);
            case 6      % RotaciÛn de -90ª entorno al eje y

                kuv(1,b)=-kuv_lamb(3);
                kuv(2,b)=kuv_lamb(2);
                kuv(3,b)=kuv_lamb(1);
                suv(1,b)=-suv_lamb(3);
```

```
                suv(2,b)=suv_lamb(2);
                suv(3,b)=suv_lamb(1);
        end

        % C·lculo de la nueva E
        E(1,1,b)=E(1,1,b)*sqrt(refl); % Mantenemos la misma polarizaciÛn.
Podriamos aleatoreizarlo tambiÈn con sqrt(I)*cos(xi) y sqrt(I)*sin(xi)
        E(1,2,b)=E(1,2,b)*sqrt(refl);
        E(2,1,b)=E(2,1,b)*sqrt(refl);
        E(2,2,b)=E(2,2,b)*sqrt(refl);

    case 'Especular'       % ReflexiÛn sÛlo
        kuv(1,b)=kuv(1,b-1)-2*dotproduct*nuv(PlanoEquivalente,1);
        kuv(2,b)=kuv(2,b-1)-2*dotproduct*nuv(PlanoEquivalente,2);
        kuv(3,b)=kuv(3,b-1)-2*dotproduct*nuv(PlanoEquivalente,3);
        LongitudK=sqrt(kuv(1,b)^2+kuv(2,b)^2+kuv(3,b)^2);
        kuv(1,b)=kuv(1,b)/LongitudK;
        kuv(2,b)=kuv(2,b)/LongitudK;
        kuv(3,b)=kuv(3,b)/LongitudK;

        % El vector s se mantiene igual.
        % Seguimos el rayo reflectado.
        E(1,1,b)=Erefl(1,1);
        E(1,2,b)=Erefl(1,2);
        E(2,1,b)=Erefl(2,1);
        E(2,2,b)=Erefl(2,2);

    case 'Refraccion'       % TransmisiÛn sÛlo
        costhetaSal=sqrt(1-(ninc/nout)^2*(1-dotproduct^2));
        kuv(1,b)=-
costhetaSal*nuv(PlanoEquivalente,1)+(ninc/nout)*(kuv(1,b-1)-
dotproduct*nuv(PlanoEquivalente,1));
        kuv(2,b)=-
costhetaSal*nuv(PlanoEquivalente,2)+(ninc/nout)*(kuv(2,b-1)-
dotproduct*nuv(PlanoEquivalente,2));
        kuv(3,b)=-
costhetaSal*nuv(PlanoEquivalente,3)+(ninc/nout)*(kuv(3,b-1)-
dotproduct*nuv(PlanoEquivalente,3));
        LongitudK=sqrt(kuv(1,b)^2+kuv(2,b)^2+kuv(3,b)^2);
        kuv(1,b)=kuv(1,b)/LongitudK;
        kuv(2,b)=kuv(2,b)/LongitudK;
        kuv(3,b)=kuv(3,b)/LongitudK;
        % Se da la vuelta al vector s, rotado por pi
        suv(1,b)=-suv(1,b);
        suv(2,b)=-suv(2,b);
        suv(3,b)=-suv(3,b);

        % Seguimiento del rayo transmitido
        E(1,1,b)=Etrans(1,1);
        E(1,2,b)=Etrans(1,2);
        E(2,1,b)=Etrans(2,1);
        E(2,2,b)=Etrans(2,2);

    case 'No change'        % n1=n2 significa no refracciÛn
        kuv(1,b)=kuv(1,b-1);
        kuv(2,b)=kuv(2,b-1);
        kuv(3,b)=kuv(3,b-1);
```

```
       % Se invierte el vector suv d·ndole la vuelta desde una parte del
plano a la otra
        suv(1,b)=-suv(1,b);
        suv(2,b)=-suv(2,b);
        suv(3,b)=-suv(3,b);

        % E permanece igual.
end

end             % Cierra el if I(b)>=Imin de
Account_for_host_absorption_loss. LÌnea 712 del cÛdigo completo.

%% WRITE BOUNCE DATA IF FINAL RAY/ESCRIBE LOS DATOS DE LOS REBOTES SI ES
EL RAYO FINAL(1340-1351) %%

if rayo==rayomax
    % call
writebouncedata(b,p,region,pos(1,b),pos(2,b),pos(3,b),kuv(1,b),kuv(3,b),ku
v(3,b),suv(1,b),suv(2,b),suv(3,b),E(0,0,b),E(0,1,b),E(1,0,b),E(1,1,b),I(b)
,w)
end
if I(b)<Imin
    break
end
end         %% ESTE ES EL END DEL FOR DEL SCRIPT "SUBSEQUENT BOUNCES AND
RAYS" ABIERTO EN LA LÕNEA 6, LÕNEA 538 DEL C”DIGO COMPLETO, LÕNEA 764 DE
LAS MACROS.

%% CALCULATE AND WRITE OUTPUTS FOR RAY/C¡LCULO Y ESCRITURA DE LAS SALIDAS
PARA RAYOS(1352-1462) %%

SumDistancia=SumDistancia+DistanciaDeRayo;
SumIPerdidoReflexExt=SumIPerdidoReflexExt+IPerdidoReflexExt;
for capa=2:capamax-1         % Omite la capa de la cÈlula y la m·xima
(externa)      % Paradoja de capamax

IPerdidoAbsPorCapaTotal=IPerdidoAbsPorCapaTotal+IPerdidoAbsPorCapa(capa);

SumIPerdidoAbsPorCapa(capa)=SumIPerdidoAbsPorCapa(capa)+IPerdidoAbsPorCapa
(capa);

SumIPerdidoAbsPorCapaTotal=SumIPerdidoAbsPorCapaTotal+IPerdidoAbsPorCapa(c
apa);
end
for pelicula=1:peliculamax      % Incluye la pelicula de la cÈlula
% Paradoja de peliculamax

IPerdidoAbsPorPeliculaTotal=IPerdidoAbsPorPeliculaTotal+IPerdidoAbsPorPeli
cula(pelicula);

SumIPerdidoAbsPorPelicula(pelicula)=SumIPerdidoAbsPorPelicula(pelicula)+IP
erdidoAbsPorPelicula(pelicula);

SumIPerdidoAbsPorPeliculaTotal=SumIPerdidoAbsPorPeliculaTotal+IPerdidoAbsP
orPelicula(pelicula);
end

for p=1:6
```

```
    % AbsorciÛn por plano (distintos de las cÈlulas) %% SERIAN EL PLANO
0?? TENGO QUE CAMBIARLO EN TODOS SITIOS??
    IPerdidoAbsPlanoTotal=IPerdidoAbsPlanoTotal+IPerdidoAbsPlano(p);
    SumIPerdidoAbsPlano(p)=SumIPerdidoAbsPlano(p)+IPerdidoAbsPlano(p);
    SumIPerdidoAbsPlanoTotal=SumIPerdidoAbsPlanoTotal+IPerdidoAbsPlano(p);

    % TransmisiÛn por plano
    IPerdidoTransPlanoTotal=IPerdidoTransPlanoTotal+IPerdidoTransPlano(p);

SumIPerdidoTransPlano(p)=SumIPerdidoTransPlano(p)+IPerdidoTransPlano(p);

SumIPerdidoTransPlanoTotal=SumIPerdidoTransPlanoTotal+IPerdidoTransPlano(p
);

    % AbsorciÛn de las cÈlulas
    IGanadoAbsPlanoTotal=IGanadoAbsPlanoTotal+IGanadoAbsPlano(p);
    SumIGanadoAbsPlano(p)=SumIGanadoAbsPlano(p)+IGanadoAbsPlano(p);
    SumIGanadoAbsPlanoTotal=SumIGanadoAbsPlanoTotal+IGanadoAbsPlano(p);
end

IRestante=Iinicial-IPerdidoReflexExt-IPerdidoAbsPorCapaTotal-
IPerdidoAbsPorPeliculaTotal-IPerdidoAbsPlanoTotal-IPerdidoTransPlanoTotal-
IGanadoAbsPlanoTotal;
SumIRestante=SumIRestante+IRestante;

% C·lculo de ganancias y pÈrdidas por la longitud de onda incidente
RayosPorLongitudOndaInicial(1,Bin)=RayosPorLongitudOndaInicial(1,Bin)+1;

PerdidasPorLongitudOndaInicial(1,Bin)=PerdidasPorLongitudOndaInicial(1,Bin
)+IPerdidoReflexExt;
PerdidasPorLongitudOndaInicial(2,Bin)=PerdidasPorLongitudOndaInicial(2,Bin
)+IPerdidoTransPlanoTotal;
PerdidasPorLongitudOndaInicial(3,Bin)=PerdidasPorLongitudOndaInicial(3,Bin
)+IPerdidoAbsPorCapaTotal;
PerdidasPorLongitudOndaInicial(4,Bin)=PerdidasPorLongitudOndaInicial(4,Bin
)+IPerdidoAbsPorPeliculaTotal;
PerdidasPorLongitudOndaInicial(7,Bin)=PerdidasPorLongitudOndaInicial(7,Bin
)+IPerdidoAbsPlanoTotal;
PerdidasPorLongitudOndaInicial(8,Bin)=PerdidasPorLongitudOndaInicial(8,Bin
)+IGanadoAbsPlanoTotal;
PerdidasPorLongitudOndaInicial(9,Bin)=PerdidasPorLongitudOndaInicial(9,Bin
)+IRestante;

for capa=2:MaxNumCapasUsuario-1

PerdidasPorLongitudOndaInicial(9+capa,Bin)=PerdidasPorLongitudOndaInicial(
9+capa,Bin)+IPerdidoAbsPorCapa(capa);
end

for pelicula=1:MaxNumPeliculasUsuario
    PerdidasPorLongitudOndaInicial(9+MaxNumCapasUsuario-
1+pelicula+1,Bin)=PerdidasPorLongitudOndaInicial(9+4-
1+pelicula+1,Bin)+IPerdidoAbsPorPelicula(pelicula);
end

% C·lculo de pÈrdidas Jsc
```

```
PerdidasJscPorLongitudOndaInicial(1,Bin)=PerdidasJscPorLongitudOndaInicial
(1,Bin)+IPerdidoReflexExt*IQECelulaInterpolado*qq*(0.000000001*w)/(hh*cc)/
(xUnidad*yUnidad);
PerdidasJscPorLongitudOndaInicial(2,Bin)=PerdidasJscPorLongitudOndaInicial
(2,Bin)+IPerdidoTransPlanoTotal*IQECelulaInterpolado*qq*(0.000000001*w)/(h
h*cc)/(xUnidad*yUnidad);
PerdidasJscPorLongitudOndaInicial(3,Bin)=PerdidasJscPorLongitudOndaInicial
(3,Bin)+IPerdidoAbsPorCapaTotal*IQECelulaInterpolado*qq*(0.000000001*w)/(h
h*cc)/(xUnidad*yUnidad);
PerdidasJscPorLongitudOndaInicial(4,Bin)=PerdidasJscPorLongitudOndaInicial
(4,Bin)+IPerdidoAbsPorPeliculaTotal*IQECelulaInterpolado*qq*(0.000000001*w
)/(hh*cc)/(xUnidad*yUnidad);
PerdidasJscPorLongitudOndaInicial(7,Bin)=PerdidasJscPorLongitudOndaInicial
(7,Bin)+IPerdidoAbsPlanoTotal*IQECelulaInterpolado*qq*(0.000000001*w)/(hh*
cc)/(xUnidad*yUnidad);
PerdidasJscPorLongitudOndaInicial(8,Bin)=PerdidasJscPorLongitudOndaInicial
(8,Bin)+IGanadoAbsPlanoTotal*IQECelulaInterpolado*qq*(0.000000001*w)/(hh*c
c)/(xUnidad*yUnidad);
PerdidasJscPorLongitudOndaInicial(9,Bin)=PerdidasJscPorLongitudOndaInicial
(9,Bin)+IRestante*IQECelulaInterpolado*qq*(0.000000001*w)/(hh*cc)/(xUnidad
*yUnidad);

% Escribir los datos en archivos si se desea
% if EscribirDatosEnArchivosTexto==1 || EscribirDatosEnSheets==1
    % call
WriteRayData(EscribirDatosEnArchivosTexto,EscribirDatosEnSheets,EscribirCo
lumnasEnSheets,f1,rayo,rayoX0,rayoY0,DistanciaDeRayo,w,w,IPerdidoReflexExt
,IPerdidoTransPlanoTotal,IPerdidoAbsPorCapaTotal,IPerdidoAbsPorPeliculaTot
al,IPerdidoAbsPlanoTotal,IGanadoAbsPlanoTotal,IRestante)
% end

% C·lculo y escritura de las fracciones de pÈrdidas cuando el n˙mero de
rayos alcanza la cantidad de 10,100,1000,etc
if mod(rayo,AlmacenSumIntervalos)==0       % AlmacÈn de datos
    IndiceAlmacen=IndiceAlmacen+1;

ISum=SumIPerdidoReflexExt+SumIPerdidoTransPlanoTotal+SumIPerdidoAbsPorCapa
Total+SumIPerdidoAbsPorPeliculaTotal+SumIPerdidoAbsPlanoTotal+SumIGanadoAb
sPlanoTotal+SumIRestante;
    AlmacenSum(1,IndiceAlmacen)=SumIPerdidoReflexExt;                    %
ReflexiÛn externa
    AlmacenSum(2,IndiceAlmacen)=SumIPerdidoTransPlanoTotal;              %
Plano de transmisiÛn
    AlmacenSum(3,IndiceAlmacen)=SumIPerdidoAbsPorCapaTotal;              %
AbsorciÛn por capa total
    AlmacenSum(4,IndiceAlmacen)=SumIPerdidoAbsPorPeliculaTotal;          %
AbsorciÛn por pelÌcula total
    AlmacenSum(5,IndiceAlmacen)=SumIPerdidoAbsPlanoTotal;                %
AbsorciÛn por plano total
    AlmacenSum(6,IndiceAlmacen)=SumIGanadoAbsPlanoTotal;                 %
AbsorciÛn de la cÈlula
    AlmacenSum(7,IndiceAlmacen)=SumIRestante;
    AlmacenSum(8,IndiceAlmacen)=ISum;                                    %
Aumenta con un intervalo constante

    for p=1:6
        AlmacenSum(8+p,IndiceAlmacen)=SumIPerdidoTransPlano(p);          %
TransmisiÛn del plano
    end
```

```
    for capa=2:MaxNumCapasUsuario-1         % Omite capa top que es el
ambiente exterior
        AlmacenSum(14+capa-1,IndiceAlmacen)=SumIPerdidoAbsPorCapa(capa);
% AbsorciÛn de la capa
    end
    for pelicula=1:MaxNumPeliculasUsuario
        AlmacenSum(14+MaxNumCapasUsuario-1-
1+pelicula+1,IndiceAlmacen)=SumIPerdidoAbsPorPelicula(pelicula);
    end
end

if rayo==10*AlmacenSumIntervalos            % C·lculo de desviaciones
est·ndar
    for PerdidasMec=1:(14+MaxNumCapasUsuario-1-1+pelicula+1)          %
LossMech=Perdidas mec·nicas????
        SumaCuadrados=0;
        ValorPpal=AlmacenSum(PerdidasMec,10)/10;            %
IndiceAlmacen=10 es el 100%
        for IndiceAlmacen=10:-1:2
            DifDelPpal=(AlmacenSum(PerdidasMec,IndiceAlmacen)-
AlmacenSum(PerdidasMec,IndiceAlmacen-1))-ValorPpal;
            SumaCuadrados=SumaCuadrados+DifDelPpal^2;
        end
        DesvEstand=sqrt(SumaCuadrados/9);
        SDOM=DesvEstand/sqrt(10);
        FraccionesPpals(PerdidasMec)=ValorPpal/(ISum/10);
        FraccionesSDOM(PerdidasMec)=SDOM/(ISum/10);

        % IMPORTANTE PARA C¡LCULOS Y RESULTADOS FINALES

        %
Worksheets("Convergence").Cells(4+PerdidasMec,2+ContadorCalcPpal).Value=Fr
accionesPpals(PerdidasMec);
        %
Worksheets("Convergence").Cells(4+PerdidasMec,9+ContadorCalcPpal).Value=2*
FraccionesSDOM(PerdidasMec); % 2*SDOM es el 95% Intervalo de Confianza
        AlmacenSum(PerdidasMec,1)=AlmacenSum(PerdidasMec,10);       % Por
ejemplo rayo 10 es la primera entrada para calcular los pasos de 10 rayos
    end
    IndiceAlmacen=1;
    AlmacenSumIntervalos=10*AlmacenSumIntervalos;
    ContadorCalcPpal=ContadorCalcPpal+1;        % Contador % Se inicializa
antes de lo que he escrito.
end
if rayo==10000
    disp('10.000 rayos simulados de 50.000 en:')
    Tiempo=toc
    disp('Segundos')
elseif rayo==20000
    disp('20.000 rayos simulados de 50.000 en:')
    Tiempo=toc
    disp('Segundos')
elseif rayo==30000
    disp('30.000 rayos simulados de 50.000 en:')
    Tiempo=toc
    disp('Segundos')
elseif rayo==40000
```

```
    disp('40.000 rayos simulados de 50.000 en:')
    Tiempo=toc
    disp('Segundos')
end
end % Este END cierra el for abierto en la lÌnea 343 en el apartado BEGIN
RAY TRACING. (lÌnea 567 en VB).

%%%%%%%%%%%%%%%%%%%%%%%%% Resultados de interÈs
%%%%%%%%%%%%%%%%%%%%%%%%%

for PerdidasMec=1:7
    CI95(PerdidasMec)=2*FraccionesSDOM(PerdidasMec)*100;            % Est·
en porcentaje
end

for Bin=1:221
    for i=1:19

PerdidasPorLongitudOndaInicial(i,Bin)=PerdidasPorLongitudOndaInicial(i,Bin
)*Iinc*xUnidad*yUnidad/rayo;
    end

    IincBin=0;
    for i=1:9

PerdidasJscPorLongitudOndaInicial(i,Bin)=PerdidasJscPorLongitudOndaInicial
(i,Bin)*Iinc*xUnidad*yUnidad/rayo;
        IincBin=IincBin+PerdidasPorLongitudOndaInicial(i,Bin);
    end

    for p=1:7

TransmisionPlanoPorLongitudOndaFinal(p,Bin)=TransmisionPlanoPorLongitudOnd
aFinal(p,Bin)*Iinc*xUnidad*yUnidad/rayo;
    end


GananciaCelulaSolarPorLongitudOndaInicial(1,Bin)=GananciaCelulaSolarPorLon
gitudOndaInicial(1,Bin)*Iinc*xUnidad*yUnidad/rayo;              %
EnergÌa

GananciaCelulaSolarPorLongitudOndaInicial(2,Bin)=GananciaCelulaSolarPorLon
gitudOndaInicial(2,Bin)*Iinc*xUnidad*yUnidad/rayo;              %
Intensidad

GananciaCelulaSolarPorLongitudOndaFinal(1,Bin)=GananciaCelulaSolarPorLongi
tudOndaFinal(1,Bin)*Iinc*xUnidad*yUnidad/rayo;                  %
EnergÌa

GananciaCelulaSolarPorLongitudOndaFinal(2,Bin)=GananciaCelulaSolarPorLongi
tudOndaFinal(2,Bin)*Iinc*xUnidad*yUnidad/rayo;                  %
Intensidad


EQEModulo(Bin)=GananciaCelulaSolarPorLongitudOndaInicial(2,Bin)/IincBin*hh
*cc/(qq*(LongitudDeOndaInicialBin+(Bin-0.5)*BinGrosor)*0.000000001); %
EQEModulo
end
Rayos=rayo;
```

```
DistanciaMediaRayos=SumDistancia/Rayos;     % cm
EnReflejadaAlFrente=Iinc*xUnidad*yUnidad*SumIPerdidoReflexExt/rayo;
EnTransmitidaFueraDelModulo=Iinc*xUnidad*yUnidad*SumIPerdidoTransPlanoTota
l/rayo;
EnAbsorbidaPorCapas=Iinc*xUnidad*yUnidad*SumIPerdidoAbsPorCapaTotal/rayo;
EnAbsorbidaPorPeliculas=Iinc*xUnidad*yUnidad*SumIPerdidoAbsPorPeliculaTota
l/rayo;
EnAbsorbidaPorPlacaTrasera=Iinc*xUnidad*yUnidad*SumIPerdidoAbsPlanoTotal/r
ayo;
EnAbsorbidaPorCelulasSolares=Iinc*xUnidad*yUnidad*SumIGanadoAbsPlanoTotal/
rayo;
EnRestante=Iinc*xUnidad*yUnidad*(Iinicial-
(SumIPerdidoReflexExt+SumIPerdidoTransPlanoTotal+SumIPerdidoAbsPorCapaTota
l+SumIPerdidoAbsPorPeliculaTotal+SumIPerdidoAbsPlanoTotal+SumIGanadoAbsPla
noTotal)/rayo);
EnIncidente=EnReflejadaAlFrente+EnTransmitidaFueraDelModulo+EnAbsorbidaPor
Capas+EnAbsorbidaPorPeliculas+EnAbsorbidaPorPlacaTrasera+EnAbsorbidaPorCel
ulasSolares+EnRestante;

JscReflejadaAlFrente=1000*sum(PerdidasJscPorLongitudOndaInicial(1,:));
JscTransmitidaFueraDelModulo=1000*sum(PerdidasJscPorLongitudOndaInicial(2,
:));
JscAbsorbidaPorCapas=1000*sum(PerdidasJscPorLongitudOndaInicial(3,:));
JscAbsorbidaPorPeliculas=1000*sum(PerdidasJscPorLongitudOndaInicial(4,:));
JscAbsorbidaPorPlacaTrasera=1000*sum(PerdidasJscPorLongitudOndaInicial(7,:
));
JscAbsorbidaPorCelulasSolares=1000*sum(PerdidasJscPorLongitudOndaInicial(8
,:));
JscRestante=1000*sum(PerdidasJscPorLongitudOndaInicial(9,:));
JscIncidente=JscReflejadaAlFrente+JscTransmitidaFueraDelModulo+JscAbsorbid
aPorCapas+JscAbsorbidaPorPeliculas+JscAbsorbidaPorPlacaTrasera+JscAbsorbid
aPorCelulasSolares+JscRestante;

EnergiaDeSalida=sum(GananciaCelulaSolarPorLongitudOndaFinal(2,:))*Voc*FF;

EficienciaDelModulo=100*EnergiaDeSalida/EnIncidente;
CorrienteCC=EficienciaDelModulo/(Voc*FF)*FicheroDeIntensidadDeEspectro/0.1
;

%%%%%%%%%%%%%%%%%%%%%%% Fin Resultados de interÈs
%%%%%%%%%%%%%%%%%%%%%%%

duracion=toc;
```

1.2 Calculate_rt_Amplitude_Coefficients

```
%%        FUNCI”N CALCULATE_RT_AMPLITUDE_COEFFICIENTS (2312-2363)
%%

function [rTE, rTM, tTE,
tTM]=Calculate_rt_Amplitude_Coefficients(thetai,thetat, ni, nt)

if abs(thetai)<0.001
    rTE(1)=(ni-nt)/(nt+ni);
    rTM(1)=-1*rTE(1);
    tTE(1)=1+rTE(1);
```

```
    tTM(1)=tTE(1);
else
    a=sin(thetai+thetat);
    b=sin(thetai-thetat);
    c=cos(thetai+thetat);
    d=cos(thetai-thetat);

    rTE(1)=-1*(b/a);
    rTM(1)=-1*rTE(1)*c/d;
    tTE(1)=1+rTE(1);
    tTM(1)=tTE(1)/d;
end

% Las componentes imaginarias son cero, aunque esto parece ser redundante
de momento. En VB est·n comentados, pero no hace ning¨n daÒo ponerlo y
quitamos errores.

 rTE(2)=0;
 rTM(2)=0;
 tTE(2)=0;
 tTM(2)=0;
```

1.3 Calculate_thinfilm_rAt_coefficients

```
%%        FUNCI”N CALCULATE_THINFILM_RAT_COEFFICIENTS (2366-2627)
%%

function [rTM,rTE,ATM,ATE,tTM,tTE,eta0TM, etaMTM, eta0TE,
etaMTE]=Calculate_thinfilm_rAt_coefficients(NumeroDePeliculas, PeliculaG,
RI, thetasup, wl, TransmisionDevuelta) % thetasup en RADIANES!!!

admitancia=1;  % Espacio libre en unidades gaussianas

for polarizacion=1:2        % TM=1; TE=2

    for pelicula=1:NumeroDePeliculas+1
        tf=PeliculaG(pelicula);
        if pelicula==1
            RI(1,2)=0;      % En VB RI(0,1), est· bien convertir el 1 en
2????
        else
            if RI(pelicula,2)<=0
                RI(pelicula,2)=1e-30;  % Esto provoca el fin del programa
            end
        end

        % DeterminaciÛn de theta (·ngulo de propagaciÛn)
        if pelicula==1
            thetapelicula(1,1)=thetasup; % Componente real en superstrate
            thetapelicula(1,2)=0;        % Componente imaginaria en
superstrate
        else
            [e1,f1]=ComplexDivision(RI(pelicula-1,1),-RI(pelicula-1,2),
RI(pelicula,1), -RI(pelicula,2));
            [e2,f2]=ComplexSine(thetapelicula(pelicula-1,1),
thetapelicula(pelicula-1,2));
            [e3,f3]=ComplexMultiplication(e1, f1, e2, f2);
```

```
            [thetapelicula(pelicula,1),
thetapelicula(pelicula,2)]=ComplexArcSine(e3, f3);
        end

        [e4, f4]=ComplexCosine(thetapelicula(pelicula,1),
thetapelicula(pelicula,2));

        % DeterminaciÛn de delta (par·metro del factor de fase)
        if pelicula>0 && pelicula<=NumeroDePeliculas
            e5=2*pi*tf*RI(pelicula,1)/wl;
            f5=2*pi*tf*(-RI(pelicula,2))/wl;
            [delta(1), delta(2)]=ComplexMultiplication(e4, f4, e5, f5);
        end

        % DeterminaciÛn de eta (depende de la polarizaciÛn)
        if polarizacion==1 % TM
            [eta(pelicula,1),
eta(pelicula,2)]=ComplexDivision(admitancia*RI(pelicula,1), -
admitancia*RI(pelicula,2), e4, f4);
        else
            [eta(pelicula,1),
eta(pelicula,2)]=ComplexMultiplication(admitancia*RI(pelicula,1),-
admitancia*RI(pelicula,2), e4, f4);
        end

        % CreaciÛn de matrices
        if pelicula>0 && NumeroDePeliculas
            [e6,f6]=ComplexCosine(delta(1), delta(2));
            [e7,f7]=ComplexSine(delta(1), delta(2));

            % Matriz(1,1)
            M11Real=e6;
            M11Imag=f6;

            % Matriz(1,2)
            [e8,f8]=ComplexDivision(e7, f7, eta(pelicula,1),
eta(pelicula,2));
            M12Real=-f8;
            M12Imag=e8;

            % Matriz(2,1)
            [e8,f8]=ComplexMultiplication(e7, f7, eta(pelicula,1),
eta(pelicula,2));
            M21Real=-f8;
            M21Imag=e8;

            % Matriz(2,2)
            M22Real=e6;
            M22Imag=f6;

            if pelicula==2
                matriz(1,1,1)=M11Real;
                matriz(1,1,2)=M11Imag;
                matriz(1,2,1)=M12Real;
                matriz(1,2,2)=M12Imag;
                matriz(2,1,1)=M21Real;
                matriz(2,1,2)=M21Imag;
                matriz(2,2,1)=M22Real;
```

```
                matriz(2,2,2)=M22Imag;
            end
            if pelicula>2
                nuevamatriz=ComplexMatrixMultiplication(matriz, M11Real,
M12Real, M21Real, M22Real, M11Imag, M12Imag, M21Imag, M22Imag);
                matriz(1,1,1)=nuevamatriz(1,1,1);
                matriz(1,1,2)=nuevamatriz(1,1,2);
                matriz(1,2,1)=nuevamatriz(1,2,1);
                matriz(1,2,2)=nuevamatriz(1,2,2);
                matriz(2,1,1)=nuevamatriz(2,1,1);
                matriz(2,1,2)=nuevamatriz(2,1,2);
                matriz(2,2,1)=nuevamatriz(2,2,1);
                matriz(2,2,2)=nuevamatriz(2,2,2);
            end
        end
    end

    if NumeroDePeliculas==0
        Bmatriz(1)=1;
        Bmatriz(2)=0;
        Cmatriz(1)=eta(NumeroDePeliculas+1,1);
        Cmatriz(2)=eta(NumeroDePeliculas+1,2);
    else
        [e9,f9]=ComplexMultiplication(eta(NumeroDePeliculas+1,1),
eta(NumeroDePeliculas+1,2), matriz(1,2,1), matriz(1,2,2));
        Bmatriz(1)=matriz(1,1,1)+e9;
        Bmatriz(2)=matriz(1,1,2)+f9;
        [e9,f9]=ComplexMultiplication(eta(NumeroDePeliculas+1,1),
eta(NumeroDePeliculas+1,2), matriz(2,2,1), matriz(2,2,2));
        Cmatriz(1)=matriz(2,1,1)+e9;
        Cmatriz(2)=matriz(2,1,2)+f9;
    end

    [eta0B(1),eta0B(2)]=ComplexMultiplication(eta(1,1), eta(1,2),
Bmatriz(1), Bmatriz(2));
    eta0BminusC(1)=eta0B(1)-Cmatriz(1);
    eta0BminusC(2)=eta0B(2)-Cmatriz(2);
    eta0BplusC(1)=eta0B(1)+Cmatriz(1);
    eta0BplusC(2)=eta0B(2)+Cmatriz(2);
    denom=eta0BplusC(1)^2+eta0BplusC(2)^2;
    termAbsorb=Bmatriz(1)*Cmatriz(1)+Bmatriz(2)*Cmatriz(2)-
eta(NumeroDePeliculas+1,1);

    [e10,f10]=ComplexDivision(eta0BminusC(1), eta0BminusC(2),
eta0BplusC(1), eta0BplusC(2));

    switch polarizacion
        case 1 %TM
            rTM(1)=e10;                         % Parte real del
coeficiente de amplitud
            rTM(2)=f10;                         % Parte imaginaria del
coeficiente de amplitud
            ATM=4*eta(1,1)*termAbsorb/denom;    % Coeficiente de
intensidad
        case 2 %TE
            rTE(1)=e10;
            rTE(2)=f10;
            ATE=4*eta(1,1)*termAbsorb/denom;
    end
```

```
    if TransmisionDevuelta==1
        [e11,f11]=ComplexDivision(2*eta(1,1), 2*eta(1,2), eta0BplusC(1),
eta0BplusC(2));

        switch polarizacion
            case 1 %TM
                tTM(1)=e11;
                tTM(2)=f11;

CompAdjustTangencial=cos(thetasup)/cos(thetapelicula(NumeroDePeliculas+1,1
));
                tTM(1)=tTM(1)*CompAdjustTangencial;
                tTM(2)=tTM(2)*CompAdjustTangencial;
            case 2 %TE
                tTE(1)=e11;
                tTE(2)=f11;
        end
    end
end

% Variables simples para los prÛximos c·lculos

n0=RI(1,1);
k0=0;
nm=RI(NumeroDePeliculas+1,1);
km=RI(NumeroDePeliculas+1,2);

% 1. Determina thetaT

[t1Re,t1Im]=ComplexDivision(n0, k0, nm, km);
[t2Re,t2Im]=ComplexMultiplication(t1Re, t1Im, sin(thetasup), 0);
[thetaTRe, thetaTIm]=ComplexArcSine(t2Re, t2Im);

% 2. Ajuste reverso del coeficiente para obtener el coeficiente de
pelÌcula fina

AjusteReverso=cos(thetaTRe)/cos(thetasup);
tTM(1)=tTM(1)*AjusteReverso;
tTM(2)=tTM(2)*AjusteReverso;

% 3. DeterminaciÛn de eta0 & etaM

eta0TM=n0/cos(thetasup);
eta0TE=n0*cos(thetasup);

[t3Re,t3Im]=ComplexCosine(thetaTRe, thetaTIm);
[t4Re,t4Im]=ComplexDivision(nm, km, t3Re, t3Im);
[t5Re,t5Im]=ComplexMultiplication(nm, km, t3Re, t3Im);

etaMTM=t4Re;
etaMTE=t5Re;

if n0>nm
    thCrit=arcsin(nm/n0);
    if thetasup>thCrit
        tTM(1)=0;
        tTM(2)=0;
```

```
        tTE(1)=0;
        tTE(2)=0;
    end
end
```

1.4 CalculateNewSuv

```
%%                    FUNCI”N CALCULATENEWSUV(2629-2652)
%%

function [suv,anew,bnew,cnew]=CalculateNewSuv(bounce, a, b, c, dotproduct,
nuv, PlanoEquivalente)

% C·lculo del vector rayo si el rayo es reflejado

anew=a-2*dotproduct*nuv(PlanoEquivalente,1);
bnew=b-2*dotproduct*nuv(PlanoEquivalente,2);
cnew=c-2*dotproduct*nuv(PlanoEquivalente,3);

% C·lculo del producto cruzado normalizado de los rayos incidente y
reflejado y es el vector suv del rebote b

xval=b*cnew-c*bnew;
yval=c*anew-a*cnew;
zval=a*bnew-b*anew;
denom=sqrt(xval^2+yval^2+zval^2);
suv(1, bounce)=xval/denom;
suv(2, bounce)=yval/denom;
suv(3, bounce)=zval/denom;
```

1.5 ComplexArcSine

```
%%                    FUNCI”N COMPLEXARCSINE(2056-2072)
%%

function [E, f]=ComplexArcSine(a, b)

if a<0
    a=-a;
    b=-b;
end

% RaÌz cuadrada del argumento = g+ih
g=1-a^2+b^2;
h=-2*a*b;

[j,k]=ComplexSquareRoot(g, h);
% Logaritmo del argumento = c+id
c=-b+j;
d=a+k;

[m,n]=ComplexLogarithm(c, d);
E=n;
f=-m;
```

1.6 ComplexCosine

```
 %%                    FUNCI”N COMPLEXCOSINE(2037-2040)
%%

function [E, f]=ComplexCosine(a, b)
E=cos(a)*cosh(b);
f=-sin(a)*sinh(b);
```

1.7 ComplexDivision

```
%%                    FUNCI”N COMPLEXDIVISION(2025-2028)
%%

function [E,f]=ComplexDivision(a,b,c,d)
E=(a*c+b*d)/(c^2+d^2);
f=(b*c-a*d)/(c^2+d^2);
```

1.8 ComplexLogarithm

```
%%                   FUNCI”N COMPLEXLOGARITHM(2041-2044)
%%

function [E, f]=ComplexLogarithm(a, b)
E=log(sqrt(a^2+b^2));
f=atan(b/a);
```

1.9 ComplexMatrixMultiplication

```
%%              FUNCI”N COMPLEXMATRIXMULTIPLICATION(2073-2095)
%%

function [MatrixOut]=ComplexMatrixMultiplication(MatrixIn, R11, R12, R21,
R22, I11, I12, I21, I22)

% (1,1)
[ROUTA,IOUTA]=ComplexMultiplication (MatrixIn(1,1,1), MatrixIn(1,1,2),
R11, I11);
[ROUTB,IOUTB]=ComplexMultiplication (MatrixIn(1,2,1), MatrixIn(1,2,2),
R21, I21);
MatrixOut(1,1,1)=ROUTA+ROUTB;
MatrixOut(1,1,2)=IOUTA+IOUTB;

% (1,2)
[ROUTA,IOUTA]=ComplexMultiplication (MatrixIn(1,1,1), MatrixIn(1,1,2),
R12, I12);
[ROUTB,IOUTB]=ComplexMultiplication (MatrixIn(1,2,1), MatrixIn(1,2,2),
R22, I22);
MatrixOut(1,2,1)=ROUTA+ROUTB;
MatrixOut(1,2,2)=IOUTA+IOUTB;

% (2,1)
[ROUTA,IOUTA]=ComplexMultiplication (MatrixIn(2,1,1), MatrixIn(2,1,2),
R11, I11);
```

```
[ROUTB,IOUTB]=ComplexMultiplication (MatrixIn(2,2,1), MatrixIn(2,2,2),
R21, I21);
MatrixOut(2,1,1)=ROUTA+ROUTB;
MatrixOut(2,1,2)=IOUTA+IOUTB;

% (2,2)
[ROUTA,IOUTA]=ComplexMultiplication (MatrixIn(2,1,1), MatrixIn(2,1,2),
R12, I12);
[ROUTB,IOUTB]=ComplexMultiplication (MatrixIn(2,2,1), MatrixIn(2,2,2),
R22, I22);
MatrixOut(2,2,1)=ROUTA+ROUTB;
MatrixOut(2,2,2)=IOUTA+IOUTB;
```

1.10 ComplexMultiplication

```
%%                FUNCI”N COMPLEXMULTIPLICATION(2029-2032)
%%

function [E, F]=ComplexMultiplication(a, b, c, d)

E=(a*c-b*d);
F=(b*c+a*d);
```

1.11 ComplexSine

```
%%                     FUNCI”N COMPLEXSINE(2033-2036)
%%

function [E, f]=ComplexSine(a, b)
E=sin(a)*cosh(b);
f=cos(a)*sinh(b);
```

1.12 ComplexSquareRoot

```
%%                   FUNCI”N COMPLEXSQUAREROOT(2045-2055)
%%

function [E, f]=ComplexSquareRoot(a, b)

r=sqrt(a^2+b^2);
if abs(r-a)>0.000001
    f=sqrt((r-a)/2);
    E=b/(2*f);
else
    f=0;
    E=sqrt(a);
end
```

1.13 DetermineWavelengthBin

```
%%              FUNCI”N DETERMINEWAVELENGTHBIN (1714-1725)              %%

function [Bin]=DetermineWavelengthBin(w,StartWavelengthBin,
EndWavelengthBin, BinWidth, Bins)

if w<StartWavelengthBin
    Bin=0;
end
```

```
if w>=EndWavelengthBin
        Bin=Bins+1;
    else
        Bin=round((w-StartWavelengthBin)/BinWidth);
end
```

1.14 InterpolateData

```
%%                  FUNCI”N INTERPOLATEDATA (1895-2000)
%%

function [y]=InterpolateData(a, dims, x, dim1, dim2, xCol, yCol)

% a: conjunto de datos? ;  dims: dimensiones en el conjunto de datos ;  x:
valor de x ;  y: valor de y (devuelto) ;
% dims1 & dims2 son las variables para las dimensiones 1 & 2 (si fuese
necesario)

% Asumimos que:
% 1) x aumenta monÛtonamente
% 2) no hay elementos en el conjunto de datos en los que x estÈ vacÌa
excepto para una secuencia continua de elementos vacÌos øen la parte alta
del conjunto de datos?

LL=1; % ignora entradas en el elemento 0. (No se si afectar· al resultado
en MATLAB)

switch dims
    case 2
        UL=length(a);    % UBound(a,1) en excel. LÌmite superior es igual
al tamaÒo del conjunto
    case 3
        UL=length(a{dim1});   % UBound(a,2) en excel. LÌmite superior es
igual al tamaÒo del conjunto
        %     case 4      % No hay ning˙n caso en todo el cÛdigo, si da
problemas lo suprimirÈ.
%         UL=max(a{3});       % UBound(a,3) en excel. LÌmite superior es
igual al tamaÒo del conjunto
end

while (UL-LL)~=1
    switch dims
        case 2
            xLL=a(LL,xCol);
            xUL=a(UL,xCol);
        case 3
            xLL=a{dim1}(LL,xCol);
            xUL=a{dim1}(UL,xCol);
%         case 4                                          No hace falta, no
se usa para nada
%             xLL=a(dim1,dim2,LL,xCol);
%             xUL=a(dim1,dim2,UL,xCol);
    end

    if x<=xLL
        switch dims
            case 2
                y=a(LL,yCol);
```

```
            case 3
                y=a{dim1}(LL,yCol);
%             case 4
%                 y=a(dim1,dim2,LL,yCol);
        end
        if LL==1 && x<xLL
            xOutsideArrayWarning='TRUE';
            return
        end
    end
    if x<UL
        % Todo va bien
    end
    if x>=UL
        if isempty(xUL)==0
            switch dims
                case 2
                    y=a(UL,yCol);
                case 3
                    y=a{dim1}(UL,yCol);
%                 case 4
%                     y=a(dim1,dim2,UL,yCol);
            end
            if x>xUL
                xOutsideArrayWarning='TRUE';
                return
            end
        else
            disp('Continuar, UL es un elemento vacÌo');
        end
    end

    % Continuando...

    guess=floor((LL+UL)/2);     % floor redondea el n˙mero hacia abajo
    switch dims
        case 2
            xguess=a(guess,xCol);
        case 3
            xguess=a{dim1}(guess,xCol);
%         case 4
%             xguess=a(dim1,dim2,guess,xCol);
    end
    if isempty(xguess)==1
        UL=guess;
    else
        if xguess<x
            LL=guess;
        else
            UL=guess;           % UL no puede estar por debajo de x
        end
    end
end
switch dims
    case 2
        xLL=a(LL,xCol);
        xUL=a(UL,xCol);
        yLL=a(LL,yCol);
        yUL=a(UL,yCol);
```

```
    case 3
        xLL=a{dim1}(LL,xCol);
        xUL=a{dim1}(UL,xCol);
        yLL=a{dim1}(LL,yCol);
        yUL=a{dim1}(UL,yCol);
%     case 4
%         xLL=a(dim1,dim2,LL,xCol);
%         xUL=a(dim1,dim2,UL,xCol);
%         yLL=a(dim1,dim2,LL,yCol);
%         yUL=a(dim1,dim2,UL,yCol);
end
if isempty(xLL)==0 && isempty(xUL)==0
    [y]=LinearInterpolation(xLL,yLL,xUL,yUL,x);
else
    xOutsideWarning='TRUE';
    if isempty(xLL)==0
        y=yLL;
    end
    if isempty(xUL)==0
        y=yUL;
    end
end
```

1.15 LinearInterpolation

```
%%                FUNCI”N LINEARINTERPOLATION (2003-2005)
%%

function [y]=LinearInterpolation(x1,y1,x2,y2,x)

y=y1+(x-x1)*(y2-y1)/(x2-x1);
```

1.16 LoadRIdata

```
%%                     FUNCI”N LOADRIDATA(2136-2160)
%%

function [Materialn, Materialk]=LoadRIdata(NombreMaterial,capa)

bandera=0;
if strcmp(NombreMaterial,'Air')
    fichero=load('Air.txt');
    bandera=1;
elseif strcmp(NombreMaterial,'EVA (McI09)')
    fichero=load('EVA (McI09).txt');
    bandera=1;
elseif strcmp(NombreMaterial,'EVA (Nag99)')
    fichero=load('EVA (Nag99).txt');
    bandera=1;
elseif strcmp(NombreMaterial,'Glass - borosilicate (Schott BF33)')
    fichero=load('Glass - borosilicate (Schott BF33).txt');
    bandera=1;
elseif strcmp(NombreMaterial,'Glass - clear sodalime (Rub85)')
    fichero=load('Glass - clear sodalime (Rub85).txt');
    bandera=1;
elseif strcmp(NombreMaterial,'Glass - green sodalime (Rub85)')
    fichero=load('Glass - clear sodalime (Rub85).txt');
    bandera=1;
```

```
elseif strcmp(NombreMaterial,'Glass - low iron sodalime (Pilkington)')
    fichero=load('Glass - low iron sodalime (Pilkington).txt');
    bandera=1;
elseif strcmp(NombreMaterial,'Glass - low iron sodalime (Rub85)')
    fichero=load('Glass - low iron sodalime (Rub85).txt');
    bandera=1;
elseif strcmp(NombreMaterial,'Glass - low iron Starphire (McI09)')
    fichero=load('Glass - low iron Starphire (McI09).txt');
    bandera=1;
elseif strcmp(NombreMaterial,'PMMA - cast (McI09)')
    fichero=load('PMMA - cast (McI09).txt');
    bandera=1;
elseif strcmp(NombreMaterial,'PMMA - cast (Roehm)')
    fichero=load('PMMA - cast (Roehm).txt');
    bandera=1;
elseif strcmp(NombreMaterial,'Si - amorphous (Pal85)')
    fichero=load('Si - amorphous (Pal85).txt');
    bandera=1;
elseif strcmp(NombreMaterial,'Si - crystalline (Gre08)')
    fichero=load('Si - crystalline (Gre08).txt');
    bandera=1;
elseif strcmp(NombreMaterial,'Si - crystalline (Gre95)')
    fichero=load('Si - crystalline (Gre95).txt');
    bandera=1;
elseif strcmp(NombreMaterial,'Silicone - DCC201 (McI09)')
    fichero=load('Silicone - DCC201 (McI09).txt');
    bandera=1;
elseif strcmp(NombreMaterial,'Silicone - DCC202 (McI09)')
    fichero=load('Silicone - DCC202 (McI09).txt');
    bandera=1;
elseif strcmp(NombreMaterial,'Silicone - DCC203 (McI09)')
    fichero=load('Silicone - DCC203 (McI09).txt');
    bandera=1;
elseif strcmp(NombreMaterial,'Silicone - DCC205 (McI09)')
    fichero=load('Silicone - DCC205 (McI09).txt');
    bandera=1;
elseif strcmp(NombreMaterial,'Silicone - Wacker Silgel612 (McI09)')
    fichero=load('Silicone - Wacker Silgel612 (McI09).txt');
    bandera=1;
elseif strcmp(NombreMaterial,'SiNx - PECVD (Bak11 - unpublished)')
    fichero=load('SiNx - PECVD (Bak11 - unpublished).txt');
    bandera=1;
elseif strcmp(NombreMaterial,'SiNx - PECVD (Bak11)')
    fichero=load('SiNx - PECVD (Bak11).txt');
    bandera=1;
elseif strcmp(NombreMaterial,'SiO2 - thermal (Pal85)')
    fichero=load('SiO2 - thermal (Pal85).txt');
    bandera=1;
elseif strcmp(NombreMaterial,'TiO2 - APCVD 400C (Tho08)')
    fichero=load('TiO2 - APCVD 400C (Tho08).txt');
    bandera=1;
elseif strcmp(NombreMaterial,'TiO2 - spray (Ric00)')
    fichero=load('TiO2 - spray (Ric00).txt');
    bandera=1;
elseif strcmp(NombreMaterial,'ZnO - spray pyrolysis (Gum06)')
    fichero=load('ZnO - spray pyrolysis (Gum06).txt');
    bandera=1;
end
```

```
if bandera==0
    display('Probablemente estÈ mal escrito el nombre del fichero al que
se quiere acceder')
else

    [tamano,M]=size(fichero);
    ficheron=zeros(tamano,2);
    ficherok=zeros(tamano,3);

    ficheron(:,1)=fichero(:,1);
    ficheron(:,2)=fichero(:,2);
    ficherok(:,1)=fichero(:,3);
    ficherok(:,2)=fichero(:,4);

    if tamano>2
        for i=tamano:-1:3
            if ficheron(i,1)==0
                ficheron(i,:)=[];
            end
            if ficherok(i,1)==0
                ficherok(i,:)=[];
            end
        end
    end

    [tamanon,N]=size(ficheron);
    Materialn{capa}=zeros(tamanon,2);
    for i=1:tamanon
        Materialn{capa}(i,1)=ficheron(i,1);
        Materialn{capa}(i,2)=ficheron(i,2);
    end
    [tamanok,S]=size(ficherok);
    Materialk{capa}=zeros(tamanok,2);
    for i=1:tamanok
        Materialk{capa}(i,1)=ficherok(i,1);
        Materialk{capa}(i,2)=ficherok(i,2);
    end
end
```

1.17 LoadSpectrum

```
%%                    FUNCI”N LOADSPECTRUM(1837-1892)
%%

function [PEspectro, FicheroDeIntensidadDeEspectro,
PrimeraLongitudDeOndaDeEspectro,
UltimaLongitudDeOndaDeEspectro]=LoadSpectrum(NombreEspectro)

PrimeraFila=5;
FilaMax=4000;
NumFila=1;
bandera=0;
longitud=0;
FicheroDeIntensidadDeEspectro=0;

if strcmp(NombreEspectro,'AM0')
    PEspectrof=load('AM0.txt');
    bandera=1;
```

```
elseif strcmp(NombreEspectro,'AM1-5d')
    PEspectrof=load('AM1-5d.txt');
    bandera=1;
elseif strcmp(NombreEspectro,'AM1-5g - NEW')
    PEspectrof=load('AM1-5g - NEW.txt');
    bandera=1;
elseif strcmp(NombreEspectro,'AM1-5g - OLD')
    PEspectrof=load('AM1-5g - OLD.txt');
    bandera=1;
elseif strcmp(NombreEspectro,'1961Feb08 12pm Alamosa (Mar99)')
    PEspectrof=load('1961Feb08 12pm Alamosa (Mar99).txt');
    bandera=1;
elseif strcmp(NombreEspectro,'1967May04 12pm Sacramento (Mar99)')
    PEspectrof=load('1967May04 12pm Sacramento (Mar99).txt');
    bandera=1;
elseif strcmp(NombreEspectro,'1976Jun24 12pm Phoenix (Mar99)')
    PEspectrof=load('1976Jun24 12pm Phoenix (Mar99).txt');
    bandera=1;
elseif strcmp(NombreEspectro,'1983Jul04 12pm Brownsville (Mar99)')
    PEspectrof=load('1983Jul04 12pm Brownsville (Mar99).txt');
    bandera=1;
elseif strcmp(NombreEspectro,'1985Dec06 12pm Buffalo (Mar99)')
    PEspectrof=load('1985Dec06 12pm Buffalo (Mar99).txt');
    bandera=1;
end

if bandera==0
    display('Probablemente estÈ mal escrito el nombre del fichero del
espectro al que se quiere acceder')
else
longitud=length(PEspectrof);
PrimeraLongitudDeOndaDeEspectro=PEspectrof(1,1);        % Valor de la
primera longitud de onda del fichero
UltimaLongitudDeOndaDeEspectro=PEspectrof(longitud,1);  % Valor de la
˙ltima longitud de onda del fichero

for i=1:longitud

    LongitudDeOndaCentral=PEspectrof(i,1);      % nm
    IntensidadEspectral=PEspectrof(i,2);        % W/m2/nm

    if i==1
        BandaLongitudDeOnda=PEspectrof(i+1,1)-PEspectrof(i,1);        %
Caso especial para el primer dato
    else
        if i==longitud
            BandaLongitudDeOnda=PEspectrof(i,1)-PEspectrof(i-1,1);    %
Caso especial para el ˙ltimo dato
        else
            BandaLongitudDeOnda=(PEspectrof(i+1,1)-PEspectrof(i-1,1))/2;
        end
    end


FicheroDeIntensidadDeEspectro=FicheroDeIntensidadDeEspectro+0.0001*Intensi
dadEspectral*BandaLongitudDeOnda;      % W/cm2
    PEspectro(i,1)=FicheroDeIntensidadDeEspectro;
    PEspectro(i,2)=LongitudDeOndaCentral;
```

```
end

if i>=FilaMax
    display('Es necesario aumentar el n˙mero m·ximo de filas')
end

for i=1:longitud
    PEspectro(i,1)=PEspectro(i,1)/FicheroDeIntensidadDeEspectro;        %
NormalizaciÛn del Espectro, va desde 0 hasta 1.
end

end
```

1.18 NewCalculateRAT

```
%%                  FUNCI”N NEWCALCULATERAT (2164-2310)
%%

function [Erefl, Etrans, refl, trans, absorb]=NewCalculateRAT(b, tf, RI,
thetasup, wl, I, E) % thetasup en radianes

% tf: Grosor de la pelÌcula
% Thetasup: ¡ngulo incidente en el ø"superstrate"?
% wl: Longitud de onda
% E: Vector de polarizaciÛn en coordenadas s,p. La E recibida es incidente
al plano; la E devuelta es "outcidente" al plano.
% Refl: ReflexiÛn.
% TransmisiÛn.
% AbsorciÛn.
% 1: componente real, 2: componente imaginaria

if tf>0

    [rTM,rTE,ATM,ATE,tTM,tTE,eta0TM, etaMTM, eta0TE,
etaMTE]=Calculate_thinfilm_rAt_coefficients(0,tf,RI,thetasup, wl,true); %
TransmisiÛnDeVuelta=true

    TEAdjust=sqrt(etaMTE/eta0TE);
    TMAdjust=sqrt(etaMTM/eta0TM);
    tTE(1)=tTE(1)*TEAdjust;
    tTE(2)=tTE(2)*TEAdjust;
    tTM(1)=tTM(1)*TMAdjust;
    tTM(2)=tTM(2)*TMAdjust;

    % Vector de polarizaciÛn para el rayo reflejado
    Erefl(1,1)=rTE(1)*E(1,1,b)-rTE(2)*E(1,2,b);
    Erefl(1,2)=rTE(1)*E(1,2,b)+rTE(2)*E(1,1,b);
    Erefl(2,1)=rTM(1)*E(2,1,b)-rTM(2)*E(2,2,b);
    Erefl(2,2)=rTM(1)*E(2,2,b)+rTM(2)*E(2,1,b);

    % Vector de polarizaciÛn para el rayo transmitido
    Etrans(1,1)=tTE(1)*E(1,1,b)-tTE(2)*E(1,2,b);
    Etrans(1,2)=tTE(1)*E(1,2,b)+tTE(2)*E(1,1,b);
    Etrans(2,1)=tTM(1)*E(2,1,b)-tTM(2)*E(2,2,b);
    Etrans(2,2)=tTM(1)*E(2,2,b)+tTM(2)*E(2,1,b);

    % C·lculo de la reflexiÛn por el coeficiente de amplitud reflectivo
    refl=Erefl(1,1)^2+Erefl(1,2)^2+Erefl(2,1)^2+Erefl(2,2)^2;
```

```
    % C·lculo de la absorciÛn por el coeficiente de amplitud absortivo
    fsXintensidad=E(1,1,b)^2+E(1,2,b)^2;
    fpXintensidad=E(2,1,b)^2+E(2,2,b)^2;
    absorb=fsXintensidad*ATE+fpXintensidad*ATM;

    % C·lculo de la transmisiÛn por el coeficiente de amplitud transmisivo
    trans=(Etrans(1,1)^2+Etrans(1,2)^2)+(Etrans(2,1)^2+Etrans(2,2)^2);

    refl=refl/I;
    trans=trans/I;
    absorb=absorb/I;
    transcheck=trans-refl-absorb;

else
    % Sin pelÌcula

    ni=RI(1,1);
    nt=RI(2,1);
    thetai=thetasup;
    thetac=100*pi/180;      % Radianes!

    if ni>nt
        thetac=asin(nt/ni); % ¡ngulo crÌtico
    end
    if abs(thetai)>=thetac
        % TIR
        refl=1;
        trans=0;
        absorb=0;

        % Vector de polarizaciÛn para el rayo reflejado
        Erefl(1,1)=E(1,1,b);
        Erefl(1,2)=E(1,2,b);
        Erefl(2,1)=E(2,1,b);
        Erefl(2,2)=E(2,2,b);

        % Vector de polarizaciÛn para el rayo transmitido, probablemente
superfluo
        Etrans(1,1)=0;
        Etrans(1,2)=0;
        Etrans(2,1)=0;
        Etrans(2,2)=0;
    else
        thetat=asin(ni*sin(thetai)/nt);         % Ley de Snell?

        [rTE, rTM, tTE, tTM]=Calculate_rt_Amplitude_Coefficients(thetai,
thetat, ni, nt);

        reflTE=rTE(1)^2;
        reflTM=rTM(1)^2;

        if thetai>(pi/2)
            thetai=pi-thetai;
        end
        Tadjust=sqrt((nt/ni)*(cos(thetat)/cos(thetai)));
        tTE(1)=Tadjust*tTE(1);
        tTE(2)=0;
        tTM(1)=Tadjust*tTM(1);
```

```
        tTM(2)=0;

        %transTE=(tTE(1)^2)*Tadjust+(tTE(2)^2)*Tadjust;
        %transTM=(tTM(1)^2)*Tadjust+(tTM(2)^2)*Tadjust;

        % Vector de polarizaciÛn para el rayo reflejado
        Erefl(1,1)=rTE(1)*E(1,1,b)-rTE(2)*E(1,2,b);
        Erefl(1,2)=rTE(1)*E(1,2,b)+rTE(2)*E(1,1,b);
        Erefl(2,1)=rTM(1)*E(2,1,b)-rTM(2)*E(2,2,b);
        Erefl(2,2)=rTM(1)*E(2,2,b)+rTM(2)*E(2,1,b);

        % Vector de polarizaciÛn para el rayo transmitido
        Etrans(1,1)=tTE(1)*E(1,1,b)-tTE(2)*E(1,2,b);
        Etrans(1,2)=tTE(1)*E(1,2,b)+tTE(2)*E(1,1,b);
        Etrans(2,1)=tTM(1)*E(2,1,b)-tTM(2)*E(2,2,b);
        Etrans(2,2)=tTM(1)*E(2,2,b)+tTM(2)*E(2,1,b);

%         refl=(1-0.5)*reflTE+0.5*reflTM;                    % ANTIGUA
FORMA DE CALCULARLOS
%         trans=(1-0.5)*transTE+0.5*transTM;
%         absorb=0;

        % C·lculo de la reflexiÛn por el coeficiente de amplitud
reflectivo
        refl=(Erefl(1,1)^2+Erefl(1,2)^2)+(Erefl(2,1)^2+Erefl(2,2)^2);
        absorb=0;
        trans=(Etrans(1,1)^2+Etrans(1,2)^2)+(Etrans(2,1)^2+Etrans(2,2)^2);

        refl=refl/I;
        trans=trans/I;
        absorb=absorb/I;
        transcheck=1-trans-refl-absorb;
    end
end
```

1.19 SetPlaneUnitVector

```
%%                  FUNCI”N SETPLANEUNITVECTORS(1803-1835)
%%

function [nuv]=SetPlaneUnitVector

% Fondo

nuv(1,1)=0; % Plano 1, x
nuv(1,2)=0; % Plano 1, y
nuv(1,3)=1; % Plano 1, z

% Top

nuv(2,1)=0;
nuv(2,2)=0;
nuv(2,3)=-1;

% Sur

nuv(3,1)=0;
nuv(3,2)=1;
```

```
nuv(3,3)=0;

% Norte

nuv(4,1)=0;
nuv(4,2)=-1;
nuv(4,3)=0;

% Este

nuv(5,1)=1;
nuv(5,2)=0;
nuv(5,3)=0;

% Oeste

nuv(6,1)=-1;
nuv(6,2)=0;
nuv(6,3)=0;
```

REFERENCIAS

1. Morris R. Cohen , I. E. Drabkin. "*A Source Book in Greek Science* ", 40 (3): 277-278, 1949.

2. Herbert W. Meyer. "*A History of Electricity and Magnetism*.," 63 (4) 572, 1972.

3. Bullock, Theodore H. , "*Electroreception* " , Springer, pp. 5-7, 2005.

4. IAEA. " *Energy, Electricity and Nuclear Power: Developments and Projections - 25 Years Past and Future*", 2007.

5. Borberly, A. and Kreider, J. F. "*Distributed Generation: The Power Paradigm for the New Millennium*". CRC Press, Boca Raton, FL. pp. 400, 2001.

6. World Nuclear Association. "*World Nuclear Power Reactors 2007-08 and Uranium Requirements*", 2008.

7. IAEA Safety standards series. "*Periodic Safety Review of nuclear power plants*", No SSG-25, PP 106, 2013.

8. (INIR) IAEA Services Series. " *Guidelines for Preparing and Conducting an Integrated Nuclear Infrastructure Review* ", No. 34, 2017

9. IAEA Nuclear Energy Series, "*Milestones in the Development of a National Infrastructure for Nuclear Power ",* No. NG-G-3.1 (Rev. 1)

10. Organismo Internacional de Energía Atómica (OIEA)-Agosto 2013.

11. UNESA. Sector eléctrico. Funcionamiento de las centrales eléctricas. Sitio web.

12. Energy Balances of Non-OECD Countries. Paris: International Energy Agency, 2017 Edition.

13. Spain: 2009. Review (Energy Policies of IEA Countries). Paris: International Energy Agency, 2009.

14. International Energy Agency (IEA). Key Worl Energy Statistics and Headline Energy Data.

15. Organisation for Economic Co-Operation and Development - Nuclear Energy Agency, 75 - Paris (France); 142 p; ISBN 92-64-03941-4; 2007; p. 37-41; FSC workshop proceedings; L'Hospitalet de l'Infant (Spain); 21-23 Nov 2005

16. Science on the Periphery. The spanish Reception of Nuclear Energy: an Attempt at Modernity? Minerva, 2005, Volume 43, Number 2, Page 197. Albert Presas I Puig.

17. Ministerio de energía, turismo y agenda digital. (http://www.minetad.gob.es/energia/nuclear/Centrales/Espana/Paginas/CentralesEspana.aspx)

18. AEE, 2010a. Observatorio eólico. Available at: /http://www.aeeolica.esS

19. Alvarez-Farizo, B., Hanley, N., 2002. Using conjoint analysis to quantify public preferences over the environmental impacts of wind farms. An example from Spain. Energy Policy 30, 107–116.

20. APPA, 2003. The experience of Spanish Renewable Energy Developers. Special Edition.

21. Allan, G.J., Bryden, I., McGregor, P.G., Stallard, T., Swales, J.K., Turner, K., Wallace, R., 2008. Concurrent and legacy economic and environmental impacts from establishing a marine energy sector in Scotland. Energy Policy 36 (7), 2734–2753.

22. ECOTEC Research & Consulting, Ltd., 1999. The impact of renewables on employment and economic growth. Draft Final Report: Main Report. ALTENER Project 4.1030/E/97-009.

23. Calde´ s, N., et al., Economic impact of solar thermal electricity deployment in Spain. Energy Policy (2009), doi:10.1016/j.enpol.2008.12.022.

24. F. Manzano-Agugliaro, A. Alcayde, F.G. Montoya, A. Zapata Sierra, C. Gil. Scientific production of renewable energies worldwide: An overview. Renew Sust Energy Rev, 18 (2013), pp. 134-143

25. Manzano-Agugliaro F, Carrillo-Valle J. Location of air in-leakage in power plants condensers by helium test. DYNA 2011;86(2):173–8.

26. I. Yüksel. Hydropower for sustainable water and energy development Renewable and Sustainable Energy Reviews, 14 (2010), pp. 462-469.

27. V. Devabhaktuni, M Alam, SSSR Depuru, RC Green, D Nims, C. NearSolar energy: trends and enabling technologies. Renewable and Sustainable Energy Reviews, 19 (1) (2013), pp. 555-564.

28. World Energy Outlook, International Energy Agency; 2018. ⟨http://www.world energyoutlook.org/weo2012/⟩

29. Rühle, Sven "Tabulated Values of the Shockley-Queisser Limit for Single Junction Solar Cells". Solar Energy. 130: 139–147, 2016. Krebs et. al., Energy Technology 2013 10.1002/ente.201300057.

30. R. Chenni, M. Makhlouf, T. Kerbache, A. BouzidA detailed modelling method for photovoltaic cells. Energy, 32 (2007), pp. 1724-1730.

31. The NEED Project P.O. Box 10101, Manassas, VA. 2011. www.NEED.org

32. F.G. Montoya, M.J. Aguilera, F. Manzano-Agugliaro, Renewable energy production in Spain: A review, Renew Sustain Energy Rev, 33 (2014), pp. 509-531.

33. Andrew S. Glassner "An introduction to Ray Tracing", 9-10, 1989.

34. Z. Yun, M. F. Iskander, "Ray tracing for radio propagation modeling: principles and applications", IEEE Access, vol. 3, pp. 1089-1100, 2015.

35. H. Holst, M. Winter, M. R. Vogt, K. Bothe, M. Köntges, R. Brendel, and P. P. Altermatt, "Application of a new ray tracing framework to the analysis of extended regions in Si solar cell modules," Energy Procedia 38, 86–93 (2013).

36. Krebs *et. al.*, Energy Technology (2013), 10.1002/ente.201300057.

37. Degli-Esposti V., Fuschini F., Vitucci E. M., and Falciasecca G., ''Speed- up techniques for ray tracing field prediction models,'' *IEEE Trans. Antennas Propag.*, 57 (5): 1469–1480, 2009.

38. Holst H., Winter M., Vogt M. R., Bothe K., Köntges M., Brendel R., and Altermatt P. P., "Application of a new ray tracing framework to the analysis of extended regions in Si solar cell modules," *Energy Procedia,* 38: 86–93, 2013.

39. Kuchkuda R. An introduction to ray tracing. Theoretical Foundations of Computer Graphics and CAD, Italy, 1987.

40. Mcintosh, Keith R, James N Cotsell, Jeff S Cumpston, Ann W Norris, Nick E Powell, and Barry M Ketola: "An optical comparison of silicone and EVA encapsulants for conventional silicon PV modules: A ray-tracing study". *34th IEEE Photovoltaic Specialists Conference. Philadelphia*, pp. 544–549, 2009.

41. McIntosh K. R., Cotsell J. N. ,Norris A. W., Powell N. E. and Ketola B. M., "An optical comparison of silicone and EVA encapsulants under various Spectra", *35 IEEE Photovoltaic Specialists Conference*, 269, 2010.

42. Ohl S. and Hahn G., "Increased internal quantum efficiency of encapsulated solar cells by using two- component silicone as encapsulant material, " *23rd EU PVSEC, Valencia,* pp. 2693-2697, 2008.

43. Yun Z., Iskander M. F., "Ray tracing for radio propagation modeling: Principles and applications", *IEEE Access*, 3: 1089-1100, 2015.

44. Zhang Z., Yun Z., and Iskander M. F., ''Ray tracing method for propaga- tion models in wireless communication systems,'' *Electron. Lett.*, 36 (5): 464–465, 2000.